

Naval Research Laboratory

Washington, DC 20375-5000



AD-A241 242



NRL Memorandum Report 6885

**Scheduling Link Activation
in Multihop Radio Networks
by Means of
Hopfield Neural Network Techniques**

CRAIG M. BARNHART AND JEFFREY E. WIESELTHIER

*Communication Systems Branch
Information Technology Division*

ANTHONY EPHREMIDES

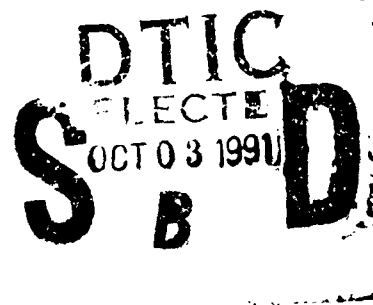
*Locus, Inc.
Alexandria, Virginia*

and

*University of Maryland
College Park, Maryland*

September 3, 1991

91-12261



Approved for public release, distribution unlimited.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 1991 September 3	3. REPORT TYPE AND DATES COVERED Interim Report 7/90 - 4/91		
4. TITLE AND SUBTITLE Scheduling Link Activation in Multihop Radio Networks by Means of Hopfield Neural Network Techniques		5. FUNDING NUMBERS PE: 61153N PR: RR021-0542 WU: DN480-557 DN159-036		
6. AUTHOR(S) Craig M. Barnhart, Jeffrey E. Wieselthier and Anthony Ephremides*				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Research Laboratory Washington, DC 20375-5000 Code 5521		8. PERFORMING ORGANIZATION REPORT NUMBER NRL Memorandum Report 6885		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research Arlington, VA 22217		10. SPONSORING / MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES *Anthony Ephremides is with the University of Maryland and Locus, Inc.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited		12b. DISTRIBUTION CODE		
13. ABSTRACT (Maximum 200 words) We address the problem of "link activation" or "scheduling" in multihop packet radio networks, a contention-free form of channel access that is appropriate for many military communication applications. It is well known that this problem, in almost all of its forms, is a combinatorial-optimization problem of high complexity. We approach this problem by the use of a Hopfield neural network model in which the method of Lagrange multipliers is used to vary dynamically the values of the coefficients used in the connection weights. Two forms of the scheduling problem are considered. In the first, communication requirements are specified in terms of the number of packets that must be transmitted over each link in the network. In the second, an additional constraint is incorporated, namely that the sequence of link activations along any multihop path must be preserved. Extensive software simulation results demonstrate the effectiveness of this approach in producing schedules of optimal length. Issues associated with the extension of this approach to the joint routing/scheduling problem are discussed.				
14. SUBJECT TERMS Communication network, Hopfield network, Multiple access, Neural network		15. NUMBER OF PAGES 100		
		16. PRICE CODE		
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UNLIMITED	

CONTENTS

1.0	INTRODUCTION	1
1.1	Outline of the Report.....	2
2.0	THE PROBLEM	3
2.1	Two Scheduling Problems: Nonsequential and Sequential Activation.....	4
2.2	Scheduling Conflicts.....	4
2.3	Communication Requirements.....	5
3.0	BOUNDS AND HEURISTICS FOR MINIMUM-LENGTH SCHEDULING.....	7
3.1	A Lower Bound on the Schedule Length.....	8
3.1.1	A Lower Bound on the Minimum Length of a Nonsequential-Activation Schedule.....	8
3.1.2	A Lower Bound on the Minimum Length of a Sequential-Activation Schedule.....	8
3.2	Heuristics for Scheduling	10
3.2.1	The NAS Heuristic.....	10
3.2.2	The SAS Heuristic	11
3.3	Performance of Scheduling Heuristics.....	11
3.3.1	Performance of the NAS Heuristic.....	11
3.3.2	Performance of the SAS Heuristic	14
4.0	A NEURAL NETWORK MODEL FOR SCHEDULING	15
4.1	Formulation of the Constraint-Energy Terms.....	19
4.1.1	The Basic Model.....	19
4.1.2	Comments on Constraint Satisfaction.....	22
4.2	Determination of Connection Weights and Bias Currents	22
4.3	Use of the Method of Lagrange Multipliers to Determine Connection Weights.....	23
4.3.1	Multiple Lagrange Multipliers.....	24
4.4	Equations of Motion	24
4.5	Variations of the Basic Scheduling NN Model	26
4.5.1	The Condensed NAS Model	26
4.5.2	The Reduced SAS Model.....	27
4.5.3	The Adjustable-Length Model	30
4.5.4	Gaussian Simulated Annealing	31
5.0	SIMULATION ISSUES	32
5.1	A Binary Interpretation of the Analog State	34
5.2	Termination Criteria.....	35
5.3	Two Methods of Evaluating NN Performance.....	36
5.4	A Modification that has Improved Simulation Results.....	36
6.0	NAS SIMULATION RESULTS.....	37
6.1	The Condensed NAS Model Using the Monte-Carlo Approach.....	38
6.2	Parameter Sensitivity.....	39
6.2.1	Bias Sensitivity	40
6.2.2	LM Time Constant Sensitivity	40
6.2.3	Sensitivity to β	42
6.3	A More Difficult NAS Problem Instance	44

6.3.1	A Third NAS Problem Instance	46
6.4	Some Improvements to the NN Model	48
6.4.1	Time-Varying β	48
6.4.2	A Traffic-Based Heuristic	49
6.5	Evaluation of the NN Improvements via the Multiple-Instance Approach.....	50
6.5.1	The Use of the Monotonic β -Function: Simulations A - C.....	51
6.5.2	The Use of the Nonmonotonic β -Function: Simulations D - E.....	51
6.5.3	The Use of a Traffic-Based Bias: Simulations E - H	52
6.5.4	NN Scheduling of Set (7, 7).....	52
6.6	An Evaluation of the Basic and the Adjustable-Length NAS Models.....	53
6.7	Conclusions on the NAS NN Model.....	56
7.0	SAS SIMULATION RESULTS	57
7.1	Sequential-Activation Scheduling with the Monte-Carlo Approach	57
7.2	Skewed Initialization.....	60
7.3	Skewed Randomization.....	63
7.4	Evaluation of Skewed Initialization and/or Randomization via Monte-Carlo Simulation.....	65
7.5	Evaluation of Skewed Initialization and/or Randomization via Multiple-Instance Simulation	67
7.5.1	SAS Multiple-Instance Simulation Issues	67
7.5.2	Evaluation of the Conditionally-Fixed-Value and the Hybrid Form of Initialization.....	68
7.5.3	Evaluation of the Path-Length-Dependent Form of Initialization	71
7.5.4	SAS of the Problem Instances with Known Minimum Schedule Lengths of 8 Slots.....	72
7.6	Conclusions on the SAS NN Model	74
8.0	THE JOINT ROUTING-SCHEDULING PROBLEM.....	74
8.1	Problem Formulation	78
8.2	A Joint Routing-Scheduling NN Model	78
8.3	Alternate Models.....	83
9.0	CONCLUSIONS.....	83
	REFERENCES.....	85
	APPENDIX A - TABLES OF THE PATHS AND PROBLEM INSTANCES ASSOCIATED WITH THE NETWORK OF FIGURE 2.1	88
	APPENDIX B - TIGHTENING THE SAS BOUND (AN EXAMPLE)	94

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

SCHEDULING LINK ACTIVATION IN MULTIHOP RADIO NETWORKS BY MEANS OF HOPFIELD NEURAL NETWORK TECHNIQUES

1.0 INTRODUCTION

In this report, we address the problem of "link activation" or "scheduling" in multihop packet radio networks [1, 2, 3]. In multihop wireless networks, channel access rules are based on either contention-based protocols or interference-free scheduled transmissions. For a variety of reasons discussed in [2] and [4], the use of interference-free scheduled transmissions is a preferable mode of access in many applications. Under this channel-access mechanism, the nodes are assigned non-interfering, periodically-recurring, time slots in which to transmit their packets. In generating these transmission schedules, it is possible to take advantage of the spatial separation of the nodes, thus permitting two nodes separated by a sufficiently large distance to transmit simultaneously.

The first problem considered in connection with this approach has been the determination of schedules of minimum length that satisfy the specified end-to-end communication requirements between a number of source-destination (SD) pairs in the network. It is by now well known that this problem, in almost all of its forms, is a combinatorial-optimization problem of high complexity (NP-complete) [5, 6, 7]. In problems of this type, the communication requirements are normally specified simply in terms of the number of packets that must be transmitted over each physical link¹ in the network. We also consider the more-difficult case in which the sequence of link activations along any multihop path in the schedule must be preserved; i.e., the case in which, for each path, the link emanating from the source must be activated first, the next link second, and so on. This problem is shown to be NP-complete, and a heuristic for a variation of the problem in wireline (i.e., nonradio) networks is presented in [8]. The advantage of sequential link activation is that it reduces end-to-end delay in the network. We also discuss extensions of our model to the joint routing-scheduling problem. Although the issues of routing and scheduling in multihop packet radio networks are highly interdependent, few studies have addressed them jointly (see e.g., [6, 7, 9]).

¹ In this report, we refer to the communication channel between two nodes as a physical link. A link corresponds to a "virtual" link that corresponds to a single unit of traffic that must traverse a given physical link. Thus, if n units of traffic are to be delivered on the physical link that connects nodes i and j , there are n parallel "links" between nodes i and j .

Because of the complexity of scheduling problems, heuristics are generally used to produce suboptimal link-activation schedules. An alternate approach to the link-activation problem is the use of a Hopfield neural network (NN) [10] to generate good, although not necessarily optimal, communication schedules [11]. Under this approach, the scheduling problem is transformed into a graph-coloring problem, with the objective of determining a coloring of the graph that requires the minimum number of colors, where each color corresponds to a time slot. In this report, we present an improved formulation of a Hopfield NN model for the scheduling problem, in which the method of Lagrange multipliers is used to vary dynamically the values of the coefficients used in the connection weights of the NN. Extensive simulation results demonstrate the effectiveness of the model. Portions of this study are also discussed in [12, 13, 14].

In an earlier phase of this study, Hopfield NN models were developed for a routing problem in which congestion is minimized in multihop radio networks [15, 16, 17]. The high degree of success achieved was, in fact, a major factor in our application of this approach to scheduling problems as well.

1.1 Outline of the Report

In Section 2 we define the two versions of the scheduling problem that are addressed in this report, i.e., the nonsequential- and sequential-activation problems. Scheduling conflicts are defined, and the multihop radio network that has served as the testing ground for our simulations is introduced.

In Section 3 we present lower bounds on the length of schedules that satisfy communication requirements for both versions of the problem, as well as heuristics for the determination of short (although not necessarily minimum-length) schedules. These bounds and heuristics are helpful in assessing the performance of the NN model.

In Section 4 we define our NN model for the scheduling problem. Constraints are established that reflect the desired behavior of the NN, i.e., the generation of schedules of minimum length. These constraints are expressed in the form of energy terms in the Lyapunov energy function, thus permitting the determination of the corresponding connection weights and bias currents, which in turn leads to the equations of motion that characterize system evolution. A key feature of our model, which has been crucial to its high degree of success, is the use of the method of Lagrange multipliers to vary the connection weights dynamically as the system state evolves. Several variations of the NN model are discussed.

In Section 5 we discuss issues associated with the simulation process. These include the generation of initial NN states, the interpretation of system state, termination criteria, and methods to evaluate performance.

In Section 6 we discuss simulation results for the nonsequential-activation scheduling (NAS) problem. The ability of our NN model to find optimum schedules for a number of diverse problem instances, including highly constrained ones, is demonstrated by means of simulations of several variations of the model. These simulations also reveal that the NN model is relatively insensitive to variations in the parameter values and the communication requirements. Modifications that further enhance the NN performance are developed and evaluated.

In Section 7 we discuss simulation results for the sequential-activation scheduling (SAS) problem. Simulations of our SAS NN model indicate that the SAS problem is significantly more difficult than the NAS problem. Modifications to the model, based on conventional heuristic methods, are implemented to give a model that generates optimal, or nearly optimal, schedules for a number of different problem instances. The length of the optimum schedule is difficult to calculate, and, for several problem instances, has only been determined by the NN's generation of a schedule whose length matches a known lower bound.

In Section 8 we address the joint routing-scheduling problem. We demonstrate that the problems of routing and scheduling are not separable. Therefore, it would be desirable to develop a NN model (or other heuristic approach) that incorporates the interactions between routing and scheduling. We outline the components of a NN model for this problem; however, the resulting model is extremely complex, except for small problems, and has not been implemented.

Finally, in Section 9 we present our conclusions from this study.

2.0 THE PROBLEM

Given the connectivity graph of a radio communication network, a set of N_{sd} SD pairs, and a multihop path connecting each SD pair, determine a link-activation schedule of minimum length that will deliver one packet² between each source and the corresponding destination such that no scheduling conflicts occur. Before addressing the nature of scheduling conflicts, we define two versions of the scheduling problem.

² The model can easily be extended to incorporate nonunit traffic requirements.

2.1 Two Scheduling Problems: Nonsequential and Sequential Activation

In the case of nonsequential-activation scheduling (NAS), the goal is to schedule each link the specified number of times in each activation cycle, without regard to the order in which the links of any path are activated. In the more-difficult case of sequential-activation scheduling (SAS), the sequence of link activations along any multihop path must be preserved; i.e., for each path, the link emanating from the source must be activated first, the next link second, and so on. The same total traffic (in terms of the number of activations of each link per cycle) is supported under both models; however, the NAS model may result in greater end-to-end packet delay because several cycles may be needed to transport a packet from source to destination. Actually, in most cases throughput (measured in terms of packets per slot) is greater using NAS than SAS because removal of the sequentiality constraint often permits satisfaction of communication requirements in fewer slots.

2.2 Scheduling Conflicts

We distinguish three types of scheduling conflicts, i.e., "primary," "secondary," and "sequence." These are described below.

Primary Conflicts

In this report we say that a *primary conflict* occurs if:

1. A node has been scheduled to transmit and receive in the same slot; or
2. A node has been scheduled to receive from two or more nodes in the same slot; or
3. A node has been scheduled to transmit to two or more nodes in the same slot.

For rather restricted systems, in which each node has a single transmitter and a single receiver, such conflicts prevent the correct reception of a packet. In systems with multiple receivers available at each platform, and/or a capability for successful simultaneous transmission and reception, the notion of primary conflict can be redefined easily to incorporate such less-restrictive constraints.

Secondary Conflicts

We say that *secondary conflicts* occur when additional signals are transmitted in the same neighborhood as the desired receiver, although not directed to that receiver. Whether or not they are destructive depends on the nature of the signaling (i.e., coding and modulation) scheme that is being used. For example, single-channel, narrowband systems normally cannot tolerate any

secondary conflicts, unless the interfering signals are of considerably lower power than the desired signal. However, in spread-spectrum code-division multiple-access (CDMA) systems, several interfering signals transmitted on codes that are quasiorthogonal to that of the desired signal can typically be tolerated; the probability of packet error in frequency-hopping systems depends on the number of frequency bins over which the signal is hopped and on the properties of the error-control coding that is used [4]. In spread-spectrum systems that use orthogonal CDMA codes this is not a problem; any number of simultaneous transmissions can be tolerated. Since our main objective is to demonstrate the capability of the NN approach rather than the precise modeling of the interference, we assume here that such orthogonal spread-spectrum signaling is used, and thus the problem of secondary conflicts is not addressed. However, in Section 4.1.1, we show how the impact of secondary conflicts may be easily included in our model to handle the case of nonorthogonal CDMA or non-spread-spectrum systems.

Sequence Conflicts

The sequential scheduling requirement is a further restriction of the problem. We declare that a *sequence conflict* occurs if, within the schedule of activation, two or more links are activated out of order. As mentioned before, under the SAS model we require that along a SD path the links are activated in the order in which they appear in the path, starting with the link that emanates from the source node.

Thus, overall, we declare the occurrence of a scheduling conflict if there is a primary conflict, or if there is a sequence conflict. However, sequence conflicts are not addressed in the formulation of the NAS problem.

2.3 Communication Requirements

The testing ground of our approach to this problem and all the relevant simulations presented in this report are based on scheduling the delivery of one unit of traffic from each of a specified set of source nodes to a specified set of destination nodes in the 24-node network shown in Fig. 2.1. In each case a single path between each of the SD pairs is prespecified. Table 2.1 gives one example of a set of paths between ten specified SD pairs. One packet is to be delivered from each source node to each destination node in the duration of every activation cycle. The resulting communication requirements are shown in Fig. 2.2; e.g., each link represented by a single arrow corresponds to a communication requirement of one packet, each double arrow to two packets, and each triple arrow to three packets.

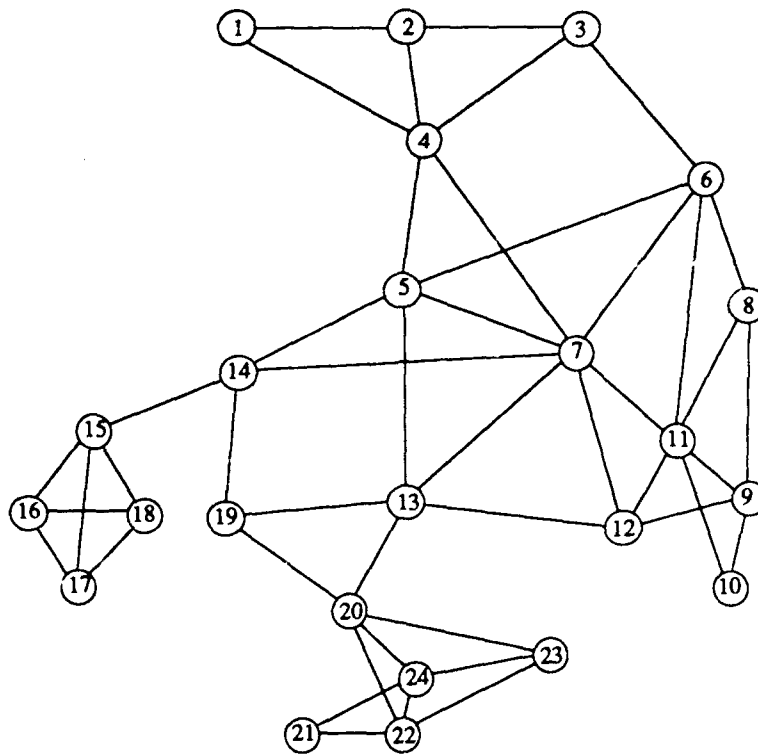


Figure 2.1. An example 24-node network

Table 2.1. A set of paths connecting 10 SD pairs in the network of Fig. 2.1

	Path					
SD pair 1: [4,24]	4	5	13	20	24	
SD pair 2: [7,17]	7	14	15	17		
SD pair 3: [9,16]	9	12	13	19	14	15 16
SD pair 4: [1,19]	1	4	5	13	19	
SD pair 5: [5,11]	5	6	11			
SD pair 6: [21,6]	21	22	20	13	5	6
SD pair 7: [1,10]	1	2	3	6	8	9 10
SD pair 8: [3,18]	3	4	7	14	15	18
SD pair 9: [2,12]	2	4	7	12		
SD pair 10: [14,8]	14	7	11	8		

Examples of primary conflicts that may occur in the network shown in Fig. 2.2 include the following: (1) If node 1 is scheduled to transmit to node 2 in the same slot in which node 2 is scheduled to transmit to node 3, node 2 is scheduled to both transmit and receive in the same slot. (2) If nodes 2 and 3 are both scheduled to transmit to node 4 at the same time, node 4 is scheduled to receive from two nodes in the same slot. (3) Node 1 is scheduled to transmit to two nodes in the same slot if it is scheduled to transmit to both nodes 2 and 4 at the same time.

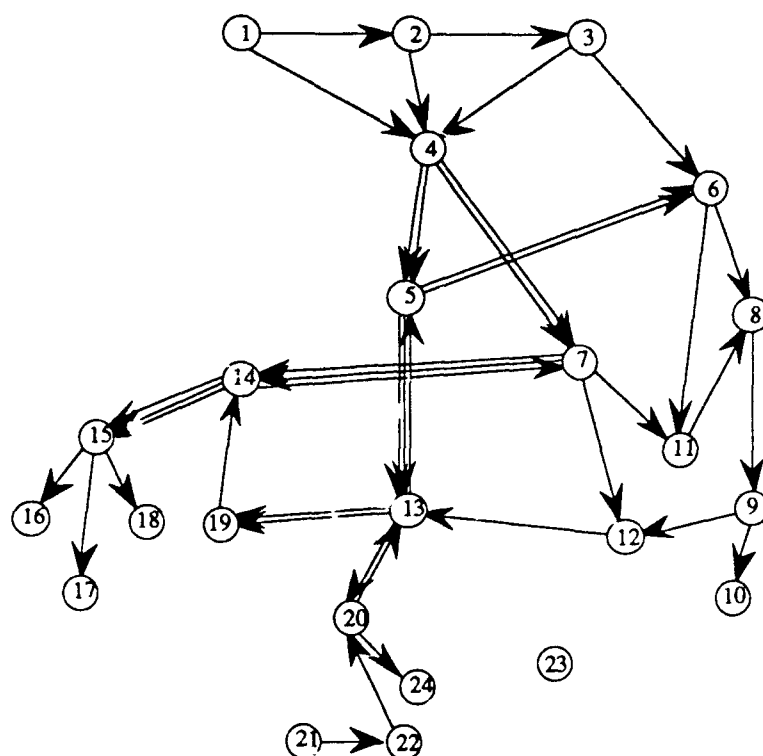


Figure 2.2. Link communication requirements for the set of paths listed in Table 2.1

An example of secondary conflict arises if there are simultaneous transmissions from node 2 to 3 and from node 1 to 4. Although the message transmitted by node 2 is intended for node 3, it also collides with node 1's transmission because node 4 is within range of node 2. Whether or not this interference is destructive depends on the type of signaling that is used. In a narrowband system, it is generally destructive. In a system with quasiorthogonal CDMA codes, it will most likely not be. However, if nodes 5 and 7 (which are also neighbors of node 4) also transmit at the same time, the combined effect of their interference may raise the packet error probability significantly.

3.0 BOUNDS AND HEURISTICS FOR MINIMUM-LENGTH SCHEDULING

It is well known that the problem of determining a minimum-length schedule that satisfies a specified end-to-end communication demand, in almost all of its forms, is NP-complete [5, 6, 7]. Therefore, unless the schedule that has been found by a NN (or by means of some heuristic algorithm) has a length equal to a known lower bound on the schedule-length, there is no way to determine whether the schedule is optimal (other than exhaustive search, which is practical only for relatively small problems). Thus, a reasonably tight lower bound on the length of the minimum-length schedule is needed to aid in determining whether a schedule meets the desired objective of being (at least) nearly minimum in length. The lower bound, on the minimum length of

nonsequential-activation schedules that are established in [6] and [3] are summarized in Section 3.1.1. The lower bound on the minimum length of sequential-activation schedules, which is introduced in Section 3.1.2, is a bound that we have developed to address the restrictions introduced by the sequential scheduling requirement.

3.1 A Lower Bound on the Schedule Length

3.1.1 A Lower Bound on the Minimum Length of a Nonsequential-Activation Schedule

For the case of nonsequential-activation scheduling (NAS), we use the lower schedule-length bound developed by Post et al. [3]. This is given by

$$B_{nas} = \max\{B_d, B_\Delta\},$$

where

B_{nas} is a lower bound on the schedule length without the sequence constraint,

$$B_d = \max_{\forall \text{ nodes } n} \{\deg(n)\},$$

$$B_\Delta = \max_{\forall \text{ nodes } n, o, p} \{f(n, o) + f(n, p) + f(o, p)\},$$

$f(n, o)$ = the number of packets that must traverse
the physical link (n, o) that connects nodes n and o .

The degree of a node n (denoted $\deg(n)$) is defined to be the sum of all flows into it plus all flows out of it. The inclusion of the expression B_Δ tightens the lower bound B_{nas} by detecting the presence of three-node cycles, which may cause the minimum schedule length Λ^* to be greater than B_d . For example, the three-node cycle shown in Fig. 3.1 clearly has $B_d = 2$, $B_\Delta = 3$, and $\Lambda^* = 3$.

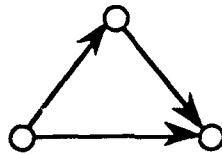


Figure 3.1. A three node cycle

3.1.2 A Lower Bound on the Minimum Length of a Sequential-Activation Schedule

For the sequential-activation scheduling (SAS) problem, we have developed a lower bound on the schedule length, denoted B_{sas} , that addresses the additional restrictions introduced by the SAS requirement. The NAS bound B_{nas} is also a lower bound on the minimum sequential-activation schedule length, which may be tightened by noting that the length of a conflict-free

sequential-activation schedule can be no shorter than the length of the longest path in the network. The observation that the positions that moderate- and high-degree nodes hold on the paths may increase the minimum schedule length is also used to tighten B_{sas} . The new bound is given by

$$B_{sas} = \max \left\{ B'_d, B_\Delta, \max_{i=1}^{N_{sd}} (L(i)) \right\},$$

where

$$B'_d = \max_{\forall \text{ nodes } n} \{ \deg(n) + F_a(n) + L_a(n) \},$$

$F_a(n)$ = The minimum number of slots required prior to the first legal activation of node n ,

$L_a(n)$ = The minimum number of slots required to complete the activation of all paths after the last activation of node n .

Since B_Δ and the maximum path length are clearly bounds for Λ^* , it suffices to show that B'_d is also a lower bound. The proof that B'_d is indeed a bound for Λ^* proceeds simply as follows.

For each node n , let \mathcal{L}_n denote the set of (unit-) traffic carrying links that include node n ; let us also refer to the j^{th} link of the path between SD pair i as the ij -link. Note that

$$F_a(n) = \left\{ \min_{ij \in \mathcal{L}_n} (j) \right\} - 1,$$

$$L_a(n) = \min_{ij \in \mathcal{L}_n} (L(i) - j),$$

where $L(i)$ is the length, in hops, of the path between SD pair i .

Clearly, in order to activate links sequentially, for each node n a minimum of $F_a(n)$ slots must be used prior to activating any of the links in \mathcal{L}_n . Additionally, a minimum of $\deg(n)$ slots are required to complete the activation of the links in \mathcal{L}_n without generating a primary conflict. Finally, an additional number of slots are needed to complete the path corresponding to the link in \mathcal{L}_n that was activated last, and this is at least $L_a(n)$. By maximizing over all nodes we get the expression for B'_d above.

Thus, B_{sas} includes some of the effects of the additional SAS constraint. Besides tightening the bound by capturing the length of the longest path, it also tightens the bound by considering those nodes of moderate and high degree that cannot be legally activated in the early

and/or later slots because of their position in the paths. For example, node 13 in Fig. 2.2 has $\deg(13) = 8$. Because its earliest appearance in any path is as a receiver in the second link in the paths between SD pairs 1 and 3 (see Table 2.1), $F_a(13) = 1$, i.e., its first legal activation can occur no earlier than the second slot. Its latest appearance is as the transmitter in the last link of the path between SD pair 4; therefore, $L_a(13) = 0$. For the network shown in the figure, the value of $B_{sas} = B_d' = \deg(13) + F_a(13) + L_a(13) = 9$ is, in fact, a tight bound on the sequential schedule length, i.e., $B_{sas} = \Lambda^*$.

3.2 Heuristics for Scheduling

To further assess the quality of NN solutions, we have also considered two forms of a “biased-greedy” heuristic similar to that developed by Post et al. in [19]. The NAS heuristic provides optimal or near-optimal nonsequential-activation schedules (schedules with length equal to or slightly greater than the NAS bound B_{nas} , respectively) for most instances of the NAS problem. The SAS heuristic generally provides sequential schedules with lengths one or two slots greater than the SAS bound B_{sas} . In Section 3.3, the performance of these heuristics is evaluated. Note that both heuristics are deterministic; thus each produces a unique set of link activations for a specific set of paths.

3.2.1 The NAS Heuristic

For the nonsequential-activation scheduling problem, the heuristic first creates a list of all the links in the network and assigns to each link a bias equal to the sum of the nodal degrees of the two nodes on which the link is incident. In this setting, a link corresponds to one unit of traffic that must traverse one hop. Thus, if four units of traffic must be passed between adjacent nodes i and j , four parallel links connect the nodes. The list of links is then sorted in descending order based upon the bias. The algorithm attempts to schedule each link in the first slot by descending through the list and activating and removing each link from the list that does not share a node with a previously activated link. When the bottom of the list is reached, the slot number is incremented, and the process is repeated; the algorithm descends through the remaining list, activating and removing each link that does not share a node with a link that was previously activated in the present slot. The process is repeated until every link has been assigned a slot and the list is empty.

3.2.2 The SAS Heuristic

The SAS heuristic, which we introduce in this report, is the first algorithm we know of for sequential link activation.³ This algorithm is essentially the same as the NAS heuristic, except that the list of links for each slot is restricted to those links that are eligible for activation in that slot, and the basis on which the links are sorted is slightly different. In the first slot, only the first links (the links emanating from a source) from each path are included in the link list because they are the only links eligible for activation. The list of links for the second slot includes only the first links that were not scheduled in the first slot and the second links in paths whose first links were scheduled in the first slot. In general, the list of links that are eligible for activation in the k^{th} slot contains no more than N_{sd} links, where N_{sd} denotes the number of SD pairs; each link in the list is less than or equal to the k^{th} link in its path, and all of the links in the path that precede a listed link have been activated in an earlier slot.

In the SAS heuristic, the bias assigned to each link is slightly altered from that used in the NAS heuristic to reflect the emphasis on sequential scheduling. Link ij (the j^{th} link in the path between SD pair i) between nodes t and r is assigned a bias given by

$$bias_{ij} = \max\{\deg(t), \deg(r)\} + \frac{L(i) - j}{\phi},$$

where ϕ , which was arbitrarily set equal to 10, may be used to shift the priority from activating nodes of high degree to activating those links furthest from the destination. The eligible links at each slot are sorted based on their bias and are “greedily” activated as in the NAS heuristic.

3.3 Performance of Scheduling Heuristics

3.3.1 Performance of the NAS Heuristic

The network shown in Fig. 2.1 is the network that was examined in [15] in conjunction with the routing-to-minimize-congestion problem. As discussed in [15], a total of 52 maximally node-disjoint paths between 10 SD pairs were found by means of Dijkstra’s algorithm. These paths are listed in Table A1 of Appendix A. There are 3,981,312 different sets of 10 single paths between each of the 10 SD pairs that may be extracted from the 52 paths listed. The NAS heuristic was applied to each of these path sets; the results are compared with the lower NAS bound B_{nas} in Fig. 3.2, which shows the fraction of paths for which schedules of the specified length were

³ In [8], Mukherji presents an algorithm for a similar problem in wire-line networks.

found. The figure shows that route selection greatly impacts the minimum obtainable schedule length. The minimum value of the bound for the nonsequential-activation schedule length ($B_{nas} = 7$ slots) was obtained for 858 path sets; the heuristic was able to find a minimum-length schedule for 481 of these 858 path sets. As noted earlier, it is not known a priori whether a schedule of length B_{nas} actually exists; however, our NAS NN model has, in fact, found 7-slot schedules for all of the path sets with $B_{nas} = 7$ (discussed in Section 6.5). At the other extreme, 256 of the path sets have minimum schedule lengths of 20 slots. The figure also shows that the heuristic schedules have lengths which are generally near B_{nas} ; 97.86% of the heuristically determined schedules were easily verified to be optimal because they have lengths equal to B_{nas} . The heuristic schedules that have lengths greater than B_{nas} exceed the bound by an average of 1.12 slots.

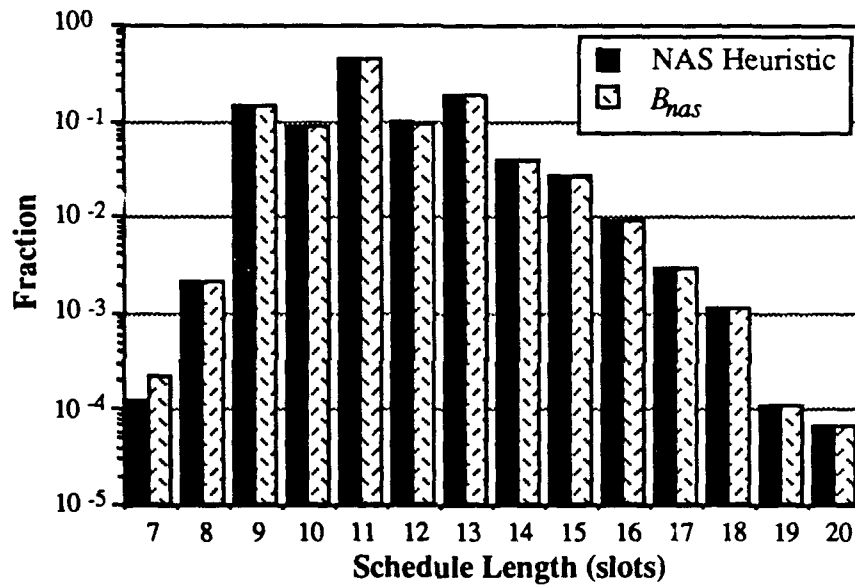


Figure 3.2. The fraction of paths for which NAS bounds and schedules of the specified length were found

The set of 858 path sets that have $B_{nas} = 7$ was divided into two disjoint subsets. The first subset consists of the 377 path sets that the NAS heuristic was unable to schedule in 7 slots. For future reference, this set is labeled “(7, >7)” ($B_{nas} = 7$, heuristic schedule length > 7). The second set consists of the 481 path sets that the NAS heuristic was able to schedule in 7 slots, and is labeled “(7, 7).” Partial listings of sets (7, >7) and (7, 7) are given in Tables A5 and A6, respectively, in Appendix A. Since all of these path sets have $B_{nas} = 7$, it is natural to wonder why the heuristic was able to find minimum-length schedules for the sets of paths in the set (7, 7), but not for those in (7, >7). It was hypothesized that the number of maximum-degree nodes in a network would give an indication of the degree of difficulty involved in finding a minimum-length conflict-free schedule. This hypothesis is partially verified by the data shown in Fig. 3.3. In this

figure, the 858 path sets with $B_{nas} = 7$ were grouped according to the number of maximum-degree nodes in each of the path sets. The black bars show the percentage of the path sets that have the number of maximum-degree nodes given by the x axis. For example, 3.5% of the path sets (30 of the 858 sets) have three degree 7 nodes, and 38.1% have five degree 7 nodes. The light bars show the percentage of each of the groups that were scheduled in 7 slots by the NAS heuristic. A total of 28 of the 30 path sets (93.3%) with three maximum-degree nodes were heuristically scheduled in 7 slots. The figure shows that as the number of maximum-degree nodes increases, the percentage of minimum-length schedules found by the NAS heuristic tends to decrease. At the extreme, the lengths of the schedules generated by the heuristic for each of the four path sets containing eight maximum-degree nodes were longer than 7 slots. Thus, the number of maximum-degree nodes in the network provides an approximate measure of the difficulty of finding a minimum-length schedule by means of the NAS heuristic.

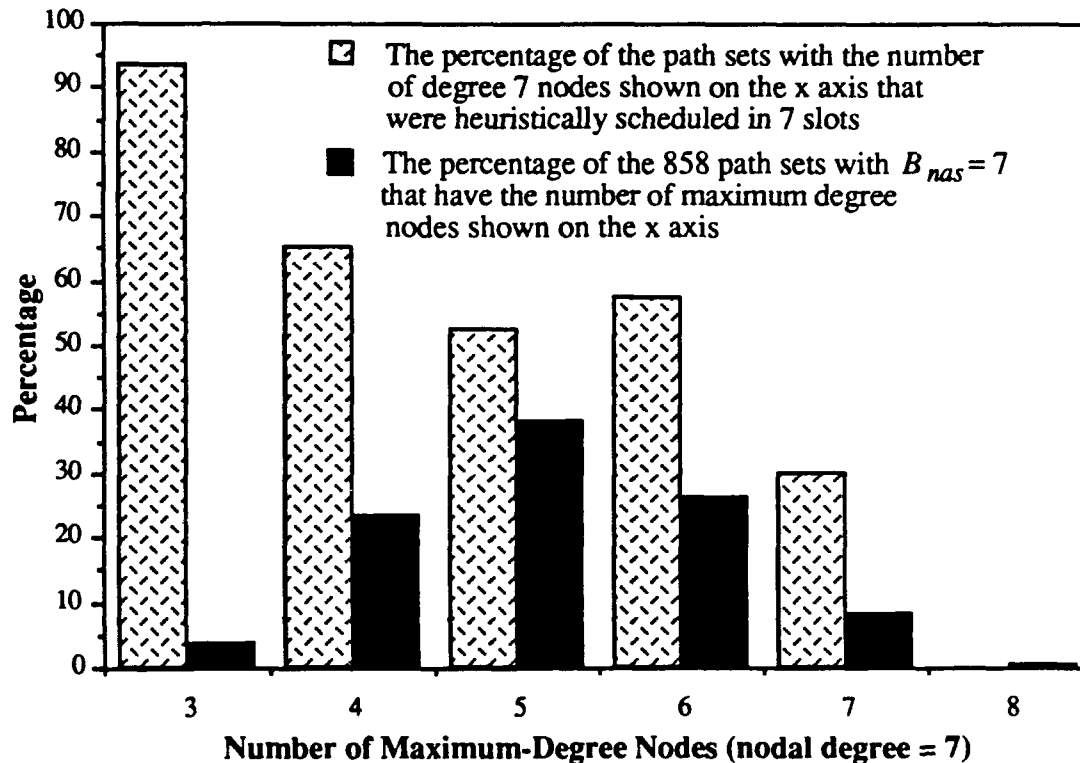


Figure 3.3. The percentage of minimum-length schedules generated by the NAS heuristic for problem instances with $B_{nas} = 7$, grouped by the number of maximum-degree nodes in the network

3.3.2 Performance of the SAS Heuristic

The SAS heuristic was similarly applied to each of the nearly 4 million different path sets, and the results are shown in Fig. 3.4. The figure shows that the SAS heuristic does not produce schedules that can easily be verified to be optimal (i.e., schedules whose length matches the bound B_{sas}) as frequently as the NAS heuristic; 31.34% of the schedules found by the SAS heuristic had lengths equal to B_{sas} , whereas 97.86% of the NAS heuristic schedule lengths matched their corresponding B_{nas} value. The SAS heuristic schedules whose length exceeded B_{sas} were an average of 1.96 hops longer than the bound. A total of 1862 sets of paths were found that had $B_{sas} = 8$, but the SAS heuristic was able to schedule only eight of these path sets in 8 slots. These eight path sets are listed in Table A4 in Appendix A. However, the apparently poor performance of the heuristic results at least partially from the looseness of the bound, B_{sas} ; that is, some (perhaps many) of the heuristically found schedules with length greater than B_{sas} may actually be minimum in length. Simulations of the SAS NN model, which are discussed in detail in Section 7, have typically yielded sequential-activation schedules with lengths between B_{sas} and the value of the heuristic schedule length. This indicates that the disparity between B_{sas} and the heuristic schedule lengths is the result of a combination of the loose bound and the inability of the SAS heuristic to find minimum-length schedules consistently.

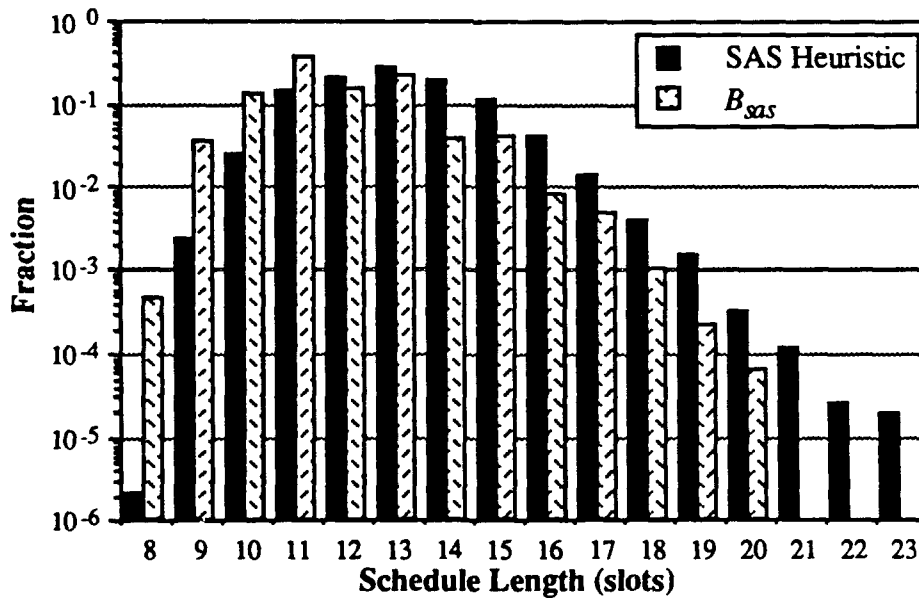


Figure 3.4. The fraction of paths for which SAS bounds and schedules of the specified length were found

4.0 A NEURAL NETWORK MODEL FOR SCHEDULING

For the reader who is not familiar with Hopfield NNs, we strongly recommend that Appendix A of [15], which provides a discussion of the use of Hopfield NNs for combinatorial-optimization problems, be read at this point to provide the necessary background material for this section. The reason that we consider such NNs here is that they have proven to be reasonably good heuristics for solving combinatorial-optimization problems.

The first step in the formulation of a Hopfield NN model is the definition of neurons that correspond to binary variables in the system that is being modeled. In this section, we consider a Hopfield NN in which, for every link in the predetermined paths connecting the N_{sd} SD pairs, one neuron is defined for each slot.⁴ For example, Fig. 4.1(a) shows a very simple six-node network with one path between each of two SD pairs, and Fig. 4.1(b) shows a possible configuration (depending on the implementation of the constraints) of the corresponding 4-slot NN model designed for the SAS problem. A triple index is used to specify the neurons, e.g., neuron ijk represents time slot k for the j^{th} link in the path connecting SD pair i . The lower horizontal plane defined by neurons 211, 231, and 234 contains all of the neurons that represent links in the path connecting SD pair 2. The parallel upper horizontal plane contains the neurons that represent the two links in the path connecting SD pair 1. Each of the parallel vertical planes defined by neurons with the same last digit corresponds to a time slot; e.g., the plane defined by neurons 211, 231, and 121 corresponds to time slot 1. The connections shown are mutually inhibitory; thus any two neurons that are directly connected to each other cannot both be "on" (i.e., have a value of 1) in a valid solution. The solid lines are present in both the NAS and SAS models, whereas the dotted lines are present in only the SAS model.⁵

The neurons are analog devices, which are characterized by an input-output relation that has the sigmoidal form

$$V_{ijk} = g(u_{ijk}) = \frac{1}{2} \left(1 + \tanh \left(\frac{u_{ijk}}{u_o} \right) \right), \quad (1)$$

where u_{ijk} and V_{ijk} are the input and output voltages, respectively, of neuron ijk , and u_o is a parameter that governs the slope of the nonlinearity. Networks of appropriately interconnected neurons of this type tend to approach an equilibrium state that, with careful modeling, will

⁴ Alternate formulations are also possible. Some of the other possibilities are discussed in Section 4.5.

⁵ Additional connections are needed to help with the satisfaction of system constraints. This figure is meant to provide a schematic representation of the principles involved in the development of a NN model, and is not meant to be complete.

correspond to a desired solution for the original problem. In our problem, in every valid solution, each link is scheduled for activation in exactly one slot. Therefore the corresponding desired equilibrium state of the NN must have exactly one of the V_{ijk} 's equal to 1 for every combination of i and j , and the others equal to 0. For example, in the NN shown in Fig. 4.1(b), exactly one of the four neurons that represent link 11 must have an output voltage of 1, which means that the link will be activated in the corresponding slot. Similarly, exactly one of the four neurons representing each of the other links must also have an output voltage of 1, and all others equal to 0. In practice, since analog neurons are used, a valid solution will have one neuron per link with an output voltage value close to 1, rather than exactly equal to 1, while the others will be close to 0. We remark here that the main advantage of the use of analog neurons is that they permit the embedding of discrete optimization problems in a continuous solution space, which results in the capability of finding good solutions much faster than is generally possible in a discrete solution space, as is discussed in Appendix A of [15].

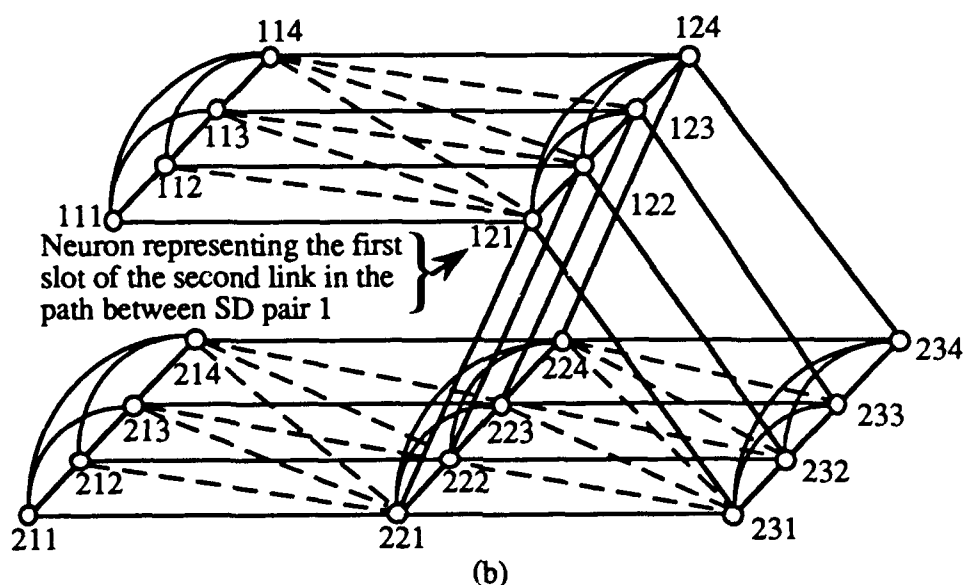
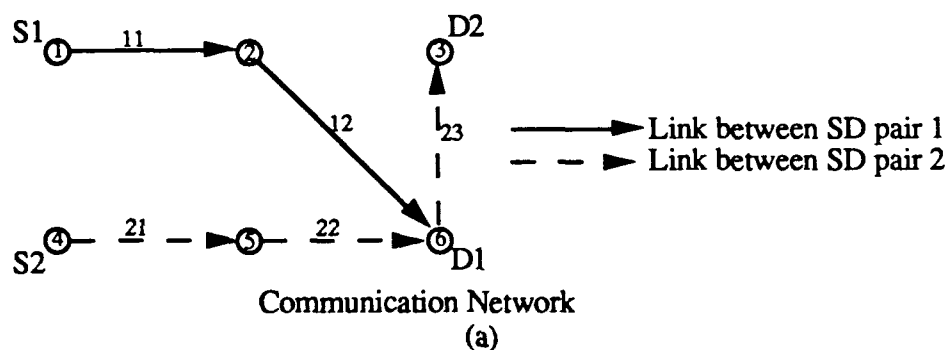


Figure 4.1. An example network, (a) shows a six-node communication network, and (b) shows the corresponding 4-slot NN model

Connections, which may be either inhibitory or excitatory, are established between all pairs of neurons. The strengths of these connections must be chosen carefully because they reflect the constraints of the optimization problem and are crucial in determining the quality of the solution. The NN evolves from some initial state to a final state that represents a local (but not necessarily global) minimum of a Lyapunov energy function, which may be written in terms of connection weights and bias currents as follows:

$$E_{total} = -\frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{l=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{m=1}^{L(m)} \sum_{k=1}^{\Lambda} \sum_{n=1}^{\Lambda} T_{ijk,lmn} V_{ijk} V_{lmn} - \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{k=1}^{\Lambda} V_{ijk} I_{ijk}. \quad (2)$$

In some applications the desired performance measure cannot be put into this form. In such cases, a Lyapunov energy function that reflects system performance goals (although possibly imprecisely) is defined. Although minimization of E_{total} in such cases does not guarantee minimization of the desired performance measure, a carefully chosen Lyapunov energy function often provides good performance.

System evolution starts from an arbitrary initial state, and follows a trajectory along which E_{total} is monotonically decreasing. In Eq. (2), $T_{ijk,lmn}$ is the strength of the connection between neurons ijk and lmn (it is positive if the connection is excitatory, and negative if it is inhibitory), I_{ijk} is the bias current applied to neuron ijk , Λ is the number of slots, and $L(i)$ is the length of the path between SD pair i in number of hops. The total number of neurons, N , is given by

$$N = \Lambda \sum_{i=1}^{N_{sd}} L(i).$$

Thus an $N \times N$ connectivity matrix \mathbf{T} can be defined, whose elements are the connection weights $T_{ijk,lmn}$, which specify the strength of the connection between neurons ijk and lmn . Convergence to a stable state is guaranteed as long as the connections are symmetric (i.e., $T_{ijk,lmn} = T_{lmn,ijk}$) [10], a condition that is satisfied by our model.

In our problem, the strengths of these connections are chosen to enforce certain constraints, which can be expressed as

$$E_c(V_1, V_2, V_3, \dots, V_N) = 0, \quad c = 1, 2, 3, \dots, C,$$

where C is the number of such constraints, and the neurons are renumbered (only in this equation) from 1 to N for convenience of notation. The specific forms of the constraints associated with our

problem are discussed in Section 4.1. By rearranging its terms, we can rewrite the energy function (Eq. (2)) as follows:

$$E_{total} = bE_{obj} + \sum_{c=1}^C \lambda_c E_c - I \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{k=1}^{\Lambda} V_{ijk}, \quad (3)$$

where E_{obj} is the objective function that we want to minimize, and which is appropriately expressed in terms of the V_{ijk} 's and $T_{ijk,lmn}$'s, the E_c 's are the constraint-energy terms, the λ_c 's are Lagrange multipliers that serve to prioritize constraint enforcement, and b is typically a constant-valued coefficient that weights minimization of the objective function. In the third term of Eq. (3), I represents an additional bias current that is applied equally to all neurons to help with satisfaction of the system constraints. Note that although E_{total} is monotonically decreasing, E_{obj} may take increasing excursions. It is important to note that the NN minimizes (locally) E_{total} while we are interested in minimizing E_{obj} . If, at equilibrium, the constraints are indeed satisfied it is hoped that the local minimum of E_{total} is also (at least) a local minimum of E_{obj} .

Typically, in applications of Hopfield-type NNs, the λ_c 's are constants whose best values are determined by trial and error. We have found, however, that by allowing the λ_c 's to vary dynamically along with the system state as in the classical method of Lagrange multipliers, we obtain significantly better NN performance. Therefore, this method is used in all of the cases studied in this report.

The NN is "programmed" by implementing the set of connection weights and bias currents that correspond to the function E_{total} that is to be minimized. An analog hardware implementation of a Hopfield NN will normally converge to its final state within at most a few RC time constants, thus providing an extremely rapid solution to a complex optimization problem. In our studies (as in most studies of this technique) we have simulated the system dynamics in software. Although such software solutions are extremely time consuming, they verify the soundness of the use of the Hopfield NN approach for optimization problems of this type and suggest that hardware implementations may be worthwhile. In fact, hardware implementation may be feasible for sizes of the problem that exceed by far the ones that can be handled in software.

It is customary and advantageous in certain cases to alter somewhat the approach to the minimization problem. Instead of trying to minimize the schedule length directly, for example, we may ask a series of binary questions like: "Is there a schedule of length Λ that satisfies all constraints (of conflict-free transmissions, here)?" for $\Lambda = \Lambda_0, \Lambda_0 + 1, \dots$, where Λ_0 is set equal

to the appropriate lower bound on the schedule length, i.e., B_{sas} for sequential-, or B_{nas} for nonsequential-activation scheduling.

Typically, a number of runs are performed from different initial states of the NN. If a schedule of Λ length cannot be found, Λ is incremented by one and the process is repeated. In this formulation of the link-activation problem, for each value of Λ there is no objective function E_{obj} to be minimized. Since there is no objective function to be minimized directly in the modified, constraint-based NN model, the energy function given by Eq. (3) may now be rewritten as:

$$E_{total} = \sum_{c=1}^4 \lambda_c E_c - I \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{k=1}^{\Lambda} V_{ijk}. \quad (4)$$

The goal is simply to determine the existence of a schedule of given length that does not violate a number of constraints, which are established to prevent transmissions that would result in collisions, and, in the case of sequential-activation scheduling (SAS), to prevent scheduling the transmission of a packet before it is received. The achievement or not of the minimum length depends on how successful the NN is in satisfying these constraints for the different values of Λ .

4.1 Formulation of the Constraint-Energy Terms

We have studied several versions of the constraints for this problem. In this section we start by presenting a "basic" version of the set of constraint formulations; then in Section 4.5 the variations of the basic version that have yielded improved performance are examined.

4.1.1 The Basic Model

Each of the constraints generates a term in Eq. (4) that must be equal to zero when the constraint is satisfied. This is simply the usual Lagrange multiplier method for constrained optimization.

Constraint 1 — (a) *Activate no links that cause primary conflicts:*

$$E_1 = \frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{l=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{m=1}^{L(l)} \sum_{k=1}^{\Lambda} |A_{ij} \cap A_{lm}| V_{ijk} V_{lmk} = 0,$$

where A_{ij} denotes the j^{th} link in the path between SD pair i , and

$$|A_{ij} \cap A_{lm}| = \begin{cases} 1, & \text{if } ij \neq lm, \text{ and links } A_{ij} \text{ and } A_{lm} \text{ share 1 or 2 nodes} \\ 0, & \text{if } ij = lm, \text{ or links } A_{ij} \text{ and } A_{lm} \text{ are disjoint} \end{cases}$$

The implementation of this constraint provides strictly inhibitory contributions to the connection weights: Two conflicting neurons (i.e., two neurons that represent adjacent links in a common slot) mutually exert on each other a negative force that is proportional to the product of their output voltages.

(b) *Limit the number of secondary conflicts:*

The effect of secondary conflicts is typically characterized by a threshold T_{sec} ; acceptable communication quality is achieved as long as the number of interfering signals in a region does not exceed this threshold. The value of T_{sec} depends strongly on signaling (i.e., waveform and modulation) considerations. For example, when only a single narrowband channel is available, no secondary interference can be tolerated, and the threshold T_{sec} is equal to 0. At the other extreme, when an orthogonal signaling scheme is used, secondary conflicts do not cause destructive interference, so T_{sec} is equal to ∞ . The most difficult case to model is that of a quasiorthogonal CDMA scheme, for which the value of T_{sec} is chosen so that the packet-error probability does not exceed a specified value. In this case, we can define $E_{sec-ijk}$, the secondary-conflict energy associated with neuron ijk , by means of a simple modification of constraint 1(a) as follows:

$$E_{sec-ijk} = \frac{1}{2} \sum_{l=1}^{N_{sd}} \sum_{m=1}^{L(l)} |A_{ij} \cap_2 A_{lm}| V_{ijk} V_{lmk} \leq T_{sec}, \quad \forall ijk$$

where

$$|A_{ij} \cap_2 A_{lm}| = \begin{cases} 1, & \text{if links } A_{ij} \text{ and } A_{lm} \text{ are node disjoint and} \\ & r_{ij} \text{ is within range of } t_{lm} \text{ or } r_{lm} \text{ is within range of } t_{ij}, \\ 0, & \text{otherwise} \end{cases}$$

r_{ij} = the receiver node of link A_{ij} , and t_{ij} = the transmit node of link A_{ij} .

One way to implement this inequality constraint would be through the use of slack variables [20]. Such an approach was taken in [21], where an additional neuron was defined for each slack variable. An alternative approach would be to view E_{sec} as part of the objective function (Eq. (3)), where

$$E_{sec} = \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{k=1}^{\Lambda} E_{sec-ijk}.$$

Under such a formulation, each secondary conflict would represent a contribution to a penalty function; thus the occurrence of secondary conflicts would be discouraged, although not prevented by any hard constraint.

For simplicity, we assume here that an orthogonal signaling scheme is being used ($T_{sec} = \infty$; thus this constraint is not implemented), since our main objective is to demonstrate the capability of the NN approach rather than the precise modeling of the interference.

Constraint 2 — *Activate each link once and only once:*

$$E_2 = \frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \left(\sum_{k=1}^{\Lambda} V_{ijk} - 1 \right)^2 = 0.$$

This term is zero when exactly one slot is chosen for each link, or, in other words, when, for every link in the network, exactly one neuron out of the set of neurons that represent different slots for that link has an output voltage of 1, and the neurons representing all of the other slots have output voltages of 0. This constraint can be either excitatory or inhibitory. Loosely speaking, the effect of this term is excitatory if the majority of links have less than one active neuron and inhibitory if the majority of links have more than one active neuron.

Constraint 3 — *Activate a total of N_x neurons:*

$$E_3 = \frac{1}{2} \left(\sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{k=1}^{\Lambda} V_{ijk} - N_x \right)^2 = 0,$$

where $N_x = \sum_{i=1}^{N_{sd}} L(i)$ (assuming one unit of traffic is to be delivered between each SD pair) is the total number of transmissions required to satisfy the communication requirements. This term vanishes when exactly one slot has been selected for each link. Like constraint 2, it can be either excitatory or inhibitory. Although this constraint would appear to be redundant (because satisfaction of the second constraint would guarantee that it is satisfied as well), its inclusion in the energy equation is helpful in achieving convergence to valid solutions. The use of such seemingly redundant constraints is common in Hopfield network models. Satisfaction of constraint 2 alone (along with a mechanism to guarantee that all neurons take on binary values) would actually suffice to constrain the number of activations. However, constraint 3 is useful because it imposes a greater penalty when an incorrect number of neurons in the entire NN are set to 1; this is because it is a quadratic form centered about N_x , whereas constraint 2 contains N_x quadratic forms each centered about 1.

Constraint 4 — *Sequentially activate the links in each path* (i.e., link ij must be activated before link im , for $m > j$):

$$E_4 = \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)-1} \sum_{m=j+1}^{L(i)} \sum_{k=1}^{\Lambda} \sum_{n=1}^{m-j+k-1} V_{ijk} V_{imn} = 0.$$

This term provides a positive contribution to the energy function whenever two neurons that represent an out-of-sequence activation of the links in a path have nonzero output voltages. It turns out that it represents purely inhibitory contributions to the connection weights. In applications where it is not necessary to maintain the sequential order of link activation, i.e., in the NAS problem, the E_4 term is not included in the energy equation.

4.1.2 Comments on Constraint Satisfaction

Since the neurons are analog devices, whose output voltages take on values in the continuum between 0 and 1, these constraints cannot be satisfied simultaneously until, and unless, a state is reached in which all output voltages take on binary values. It is possible for constraints 2 and 3 to be satisfied by a state in which more than N_x neurons are partially active (i.e., have output values less than 1). This is why, in the neuron input/output relationship, a relatively steep nonlinearity is used to force the neuron output voltages towards binary values. Thus, although the system evolves through the interior of an N -dimensional hypercube, the incorporation of these constraints into the energy function encourages the system to evolve to "legal states," which are binary states in which the constraints are, in fact, satisfied. Whether or not convergence to legal states is achieved, depends on factors such as the λ_c coefficient values, the initial state of the system, the slope of the input-output nonlinearity, and the time constants used in the iteration. All of these factors will be discussed later.

4.2 Determination of Connection Weights and Bias Currents

Substitution of the constraint-energy expressions into Eq. (4) yields:

$$E_{total} = \frac{\lambda_1}{2} \sum_{i=1}^{N_{sd}} \sum_{l=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{m=1}^{L(l)} \sum_{k=1}^{\Lambda} |A_{ij} \cap A_{lm}| V_{ijk} V_{lmk} + \frac{\lambda_2}{2} \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \left(\sum_{k=1}^{\Lambda} V_{ijk} - 1 \right)^2$$

$$+ \frac{\lambda_3}{2} \left(\sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{k=1}^{\Lambda} V_{ijk} - N_x \right)^2 + \lambda_4 \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)-1} \sum_{m=j+1}^{L(i)} \sum_{k=1}^{\Lambda} \sum_{n=1}^{m-j+k-1} V_{ijk} V_{imn} - I \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{k=1}^{\Lambda} V_{ijk}. \quad (5)$$

To determine the connection weights, we compare Eq. (5) with the generic form given in Eq. (2). The energy function contains both quadratic and linear terms. The coefficients of the quadratic terms, which involve products of the form $V_{ijk}V_{lmn}$, correspond to connection weights of the form $T_{ijk,lmn}$. Thus the connection weight $T_{ijk,lmn}$ is the sum of all coefficients that multiply the product $V_{ijk}V_{lmn}$ in Eq. (5):

$$T_{ijk,lmn} = -\lambda_1 |A_{ij} \cap A_{lm}| \delta_{kn} - \lambda_2 \delta_{ik} \delta_{jm} - \lambda_3 - \lambda_4 \delta_{il} (1 - \delta_{jm}) \cdot 1(j < m) \cdot 1(n \leq \{m - j + k - 1\}), \quad (6)$$

where δ_{ik} is the Kronecker delta symbol, and

$$1(\bullet) = \begin{cases} 1, & \text{if } \bullet \text{ is true} \\ 0, & \text{if } \bullet \text{ is false} \end{cases}$$

Similarly, the coefficients of the linear terms, which involve the V_{ijk} 's one at a time, correspond to the bias currents. Thus I_{ijk} is the sum of the coefficients that multiply V_{ijk} , which results in

$$I_{ijk} = \lambda_2 + \lambda_3 N_x + I.$$

4.3 Use of the Method of Lagrange Multipliers to Determine Connection Weights

Wacholder et al. [22] observed that the energy expression corresponding to each equality constraint may be used to update the corresponding connection coefficients by allowing the Lagrange multipliers (LM) λ_c to vary dynamically. Thus, each λ_c , at each iteration, is increased by an amount proportional to its corresponding constraint energy evaluated in the previous iteration. That is, at the $(n+1)^{\text{st}}$ iteration, we have

$$\lambda_c(n+1) = \lambda_c(n) + (\Delta t)_{\lambda_c} E_c(n),$$

where the time constant $(\Delta t)_{\lambda_c}$ may be a different value for each of the λ_c 's. Note that, since $E_c \geq 0$, the quantities λ_c are monotonically nondecreasing. Typically, the Lagrange multipliers are assigned initial values of 1.

The primary advantage of this method is that it eliminates the need to perform a trial-and-error search for the best system parameters, which is normally required for Hopfield network models. Such a search is especially time consuming in large networks because many (e.g., 100) runs with different random initial conditions are typically needed to assess the performance achievable when a particular set of parameters is used. We have used this method with a great deal

of success in our studies of routing for the minimization of congestion [15, 16]; therefore, it is used here as well.

4.3.1 Multiple Lagrange Multipliers

Examination of the second constraint energy term E_2 , which requires that each link be activated once and only once, suggests that it may be advantageous to define a separate Lagrange multiplier for the constraint applied to each link. Doing so would increase the LMs associated with those links that were unsuccessful in activating exactly one neuron. We call this the method of "multiple Lagrange multipliers" (MLM).

The second constraint formulation may be rewritten as

$$E_2 = \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} e_{2ij} = 0,$$

where each of the terms of the form

$$e_{2ij} = \frac{1}{2} \left(\sum_{k=1}^{\Lambda} V_{ijk} - 1 \right)^2 = 0$$

is an equality constraint specifically for the j^{th} link between SD pair i . Now Lagrange multipliers are defined to correspond to each of the e_{2ij} 's, and they evolve as follows,

$$\lambda_{2ij}(n+1) = \lambda_{2ij}(n) + (\Delta t)_{\lambda_{2ij}} e_{2ij}(n).$$

4.4 Equations of Motion

An equation characterizing the evolution of the input voltage at each neuron can be obtained by examining the circuit diagram of the standard Hopfield NN model shown in Fig. 4.2. The resulting expression is

$$\frac{du_{ijk}}{dt} = -\frac{u_{ijk}}{\tau} + \sum_{l=1}^{N_{sd}} \sum_{m=1}^{L(i)} \sum_{n=1}^{\Lambda} T_{ijk,lmn} V_{lmn} + I_{ijk},$$

where $\tau = RC$ (which may be set equal to 1 without loss of generality) is the time constant of the RC circuit connected to the neuron. This relationship may be expressed in terms of the energy function as:

$$\frac{du_{ijk}}{dt} = -\frac{\partial E_{total}}{\partial V_{ijk}} - \frac{u_{ijk}}{\tau}. \quad (7)$$

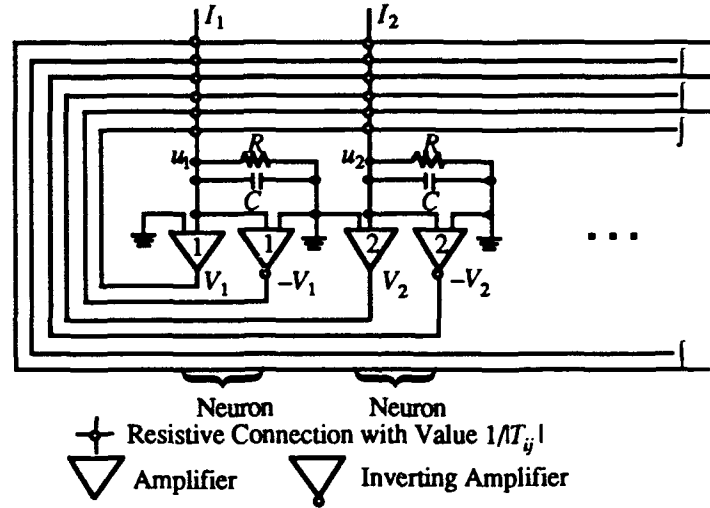


Figure 4.2. Portion of a Hopfield NN

As the system evolves from an initial state, the energy function decreases monotonically until equilibrium at a (local) minimum is reached. Since only a local minimum can be guaranteed, the final state depends on the initial state at which the system evolution is started; hence the need for the simulation of a number of runs from different initial states (random seeds). The equations of motion may be expressed in iterative form as follows:

$$\begin{aligned} u_{ijk}(t + \Delta t) = & u_{ijk}(t) - (\Delta t)u_{ijk}(t) - (\Delta t)\lambda_1 \sum_{l=1}^{N_{ad}} \sum_{m=1}^{L(l)} |A_{ij} \cap A_{lm}| V_{lmk} \\ & - (\Delta t)\lambda_2 \left(\sum_{n=1}^{\Lambda} V_{ijn} - 1 \right) - (\Delta t)\lambda_3 \left(\sum_{l=1}^{N_{ad}} \sum_{m=1}^{L(l)} \sum_{n=1}^{\Lambda} V_{lmn} - N_x \right) \\ & - (\Delta t)\lambda_4 \left((1 - \delta_{jL(i)}) \sum_{m=j+1}^{L(i)} \sum_{n=1}^{m-j+k-1} V_{imn} + (1 - \delta_{j1}) \sum_{m=1}^{j-1} \sum_{n=\max\{1, k-j+m+1\}}^{\Lambda} V_{imn} \right) + (\Delta t)I_{ijk}. \end{aligned}$$

This form is customary and appropriate for computation. A number of issues arise when these equations are simulated in software. The most apparent is the need to choose the coefficients in the weight matrix, i.e., the λ_c 's and the $(\Delta t)\lambda_c$'s. Also important is the bias current parameter I . Somewhat more subtle is the impact of the step size Δt and the nonlinearity parameter u_0 . The ability of the state to converge to an admissible solution depends strongly on these parameters. A complete discussion of these and other issues that have arisen in the simulation process is presented in Sections 5, 6, and 7. At this point, we remark that it is difficult to develop

“cookbook” procedures for the choice of specific fixed values of these parameters that will produce high-quality performance for a wide variety of networks. However, we note that we have obtained excellent and highly robust results by using the aforementioned method of Lagrange multipliers, which permits the connection weights to vary dynamically along with the evolution of the system state.

4.5 Variations of the Basic Scheduling NN Model

4.5.1 The Condensed NAS Model

Although the NAS model attempts to schedule the communication requirements for a set of multihop paths, this formulation of the problem permits a decomposition that essentially transforms it to a one-hop scheduling problem similar to that considered in [7]. As such, it is no longer necessary to associate each link activation with a particular path or SD pair. Therefore, the size of the NN model may be reduced by creating Λ neurons for each physical link, rather than creating Λ neurons for each unit of traffic on each physical link as was done in the basic model. Then constraint 2 is altered to require that each physical link p be activated $N_x(p)$ times, where $N_x(p)$ is the number of units of traffic that must traverse physical link p . We refer to this as the condensed NAS model. Denoting the modified formulation of the condensed NAS model with a superscript c , the resulting constraint may be expressed as

$$E_2^c = \frac{1}{2} \sum_{p=1}^{N_{pl}} \left(\sum_{k=1}^{\Lambda} V_{pk} - N_x(p) \right)^2 = 0,$$

where N_{pl} is the number of physical links in the network.⁶ Differentiation of E_2^c with respect to V_{pk} , as suggested by Eq. (7), yields the corresponding equation of motion term:

$$\frac{-\partial E_2^c}{\partial V_{pk}} = N_x(p) - \sum_{n=1}^{\Lambda} V_{pn}.$$

Note that now the neurons are double indexed (a triple index was needed in the basic model) so that neuron pk represents the p^{th} physical link at the k^{th} time slot. Although the other constraints remain virtually unchanged, the modified structure of the condensed NAS NN model requires that

⁶ This expression may be easily converted to the MLM format by applying an approach similar to that used in Section 4.3.1. That is, for each physical link p , a Lagrange multiplier is defined to correspond to an equality constraint that requires link p to be activated exactly $N_x(p)$ times.

each of the constraint-energy terms be rewritten to conform to the new doubly indexed neuron notation. This gives the following modified total energy equation:

$$E_{total}^c = \frac{\lambda_1}{2} \sum_{p=1}^{N_{pl}} \sum_{\substack{q=1 \\ q \neq p}}^{N_{pl}} \sum_{k=1}^{\Lambda} |A_p \cap A_q| V_{pk} V_{qk} + \frac{\lambda_2}{2} \sum_{p=1}^{N_{pl}} \left(\sum_{k=1}^{\Lambda} V_{pk} - N_x(p) \right)^2 \\ + \frac{\lambda_3}{2} \left(\sum_{p=1}^{N_{pl}} \sum_{k=1}^{\Lambda} V_{pk} - N_x \right)^2 - I \sum_{p=1}^{N_{pl}} \sum_{k=1}^{\Lambda} V_{pk},$$

where we now have

$$|A_p \cap A_q| = \begin{cases} 1, & \text{if physical links } A_p \text{ and } A_q \text{ share 1 or 2 nodes} \\ 0, & \text{if physical links } A_p \text{ and } A_q \text{ are node disjoint} \end{cases}$$

Because this is strictly a NAS model, the sequentiality constraint E_4 has been omitted.

The communication requirements of Table 2.1 represent a typical example considered in this report. Scheduling these requirements with the basic NAS model requires a NN consisting of (41 required transmissions \times 8 slots $=$) 328 neurons. The use of the condensed NAS model results in a reduction in the number of neurons to (30 physical links \times 8 slots $=$) 240. Since the number of computations required at each iteration is approximately proportional to the square of the number of neurons, this reduction in the number of neurons markedly reduces the NN complexity. Furthermore, extensive simulation results, which are discussed in Section 6, have shown that the condensed NAS model consistently delivers better performance than the basic model.

4.5.2 The Reduced SAS Model

The condensing process just discussed for the NAS model cannot be applied to the SAS model. Under the SAS operation, it is necessary to keep track of the SD pair for which each link is activated, so that the sequential order of link activations over every SD pair can be maintained. However, the number of neurons can be reduced by eliminating those neurons from consideration whose activation would not be consistent with the sequential-activation constraint. For example, the second link of a path cannot be activated in the first slot of the schedule, etc. We refer to the resulting model as the reduced SAS model.

When the sequential-activation constraint is enforced, the set of slots in which link ij may be legally activated is a subset of the set of Λ slots that form the schedule. If, for example, the path

between SD pair x has length $L(x) = 5$, and the sequential schedule length is $\Lambda = 6$, the first link of the path, link x_1 , may be legally activated only in one of the first two slots; activation in a later slot must result in either a primary conflict, a sequence conflict, or both, or the failure to schedule the activation of every link. Similarly, the second link in path x , link x_2 , may be legally activated only in one of slots 2 or 3, and link x_j may be legally activated only in one of slots j or $j + 1$. In general, link ij may be legally activated only in one of slots j through $j + X_i$, where $X_i = \Lambda - L(i)$. Therefore, the neurons that represent illegal slots (i.e., neurons ijk , $k < j$ or $k > j + X_i$) may be eliminated without reducing the admissible⁷ solution space. The elimination of these neurons, besides reducing the complexity of the NN, also removes a significant number of potential local minima that may trap the NN in an inadmissible solution.

Thus, in the reduced SAS model, only $X_i + 1$ neurons are created for each link in the path between SD pair i , rather than Λ . The constraint formulations of the basic model are virtually unchanged; only the indices of the summations over the number of slots must be changed to reflect the absence of neurons that correspond to illegal slots. (Alternatively, the removed neurons may be considered to have output voltage values of zero, in which case the indices of the basic model equations need not be altered.)

Figure 4.3 shows the reduced SAS model for the network of Fig. 4.1(a). A comparison of the model in Fig. 4.3 with that shown in Fig. 4.1(b) illustrates that, even in this very simple problem, a significant reduction in both the number of neurons, and the number of connections, is obtained through the use of the reduced SAS model. The solid lines in Fig. 4.3 represent the neuron interconnections that enforce the first two constraints. Those that are parallel to the y axis enforce the first constraint, which prohibits primary conflicts. The interconnections that are parallel to the *time* axis enforce the second constraint (activate each link exactly once). The dashed lines represent the interconnections that enforce the sequentiality constraint. The interconnections between every pair of neurons that enforce the third constraint (activate a total of N_x neurons) are not shown.

The minimum length of a sequential-activation schedule that satisfies the communication requirements of Fig. 4.1(a) is 4 slots. Table 4.1 lists one such schedule. In the table, entries are shown only for the slots that are represented by a neuron in Fig. 4.3. The blank cells represent the slots in which the link can never be activated in an admissible sequential-activation schedule. Thus, the table also aids in understanding the structure of the reduced SAS NN model. Since each of the cells in the table corresponds to a neuron, the cells are addressed by a triple index in the

⁷ An admissible solution or schedule is one that satisfies all of the constraints.

same manner as the neurons; i.e., cell i,j,k represents slot k of the j^{th} link in the path between SD pair i . A cell entry of "X" denotes a link activation, "o" denotes an open slot (i.e., the link may be activated with out conflict), "b" denotes a blocked slot in which activation will result in a primary conflict, and "i" denotes an ineligible slot (i.e., a slot in which activation of the link must cause a sequence conflict). For example, the "b" entry in cell 1,2,4 indicates that activation of link 1,2 (the second link in the path connecting SD pair 1) is blocked in the fourth slot by the scheduled activation of link 2,3 in this slot. Therefore, activation of link 1,2 in the fourth slot would cause a primary conflict. A close examination of the table reveals that link 1,1 may be activated in slot 3 without causing a primary conflict. However, because link 1,2 is blocked in slot 4, there is no way that a unit of traffic received in slot 3 can be relayed by link 1,2 in this four-slot cycle; i.e., as a result of the blockage of link 1,2 in slot 4, activation of link 1,1 in slot 3 must result in a sequence conflict. Therefore, we declare link 1,1 ineligible for activation in slot 3, and enter an "i" in cell 1,1,3.

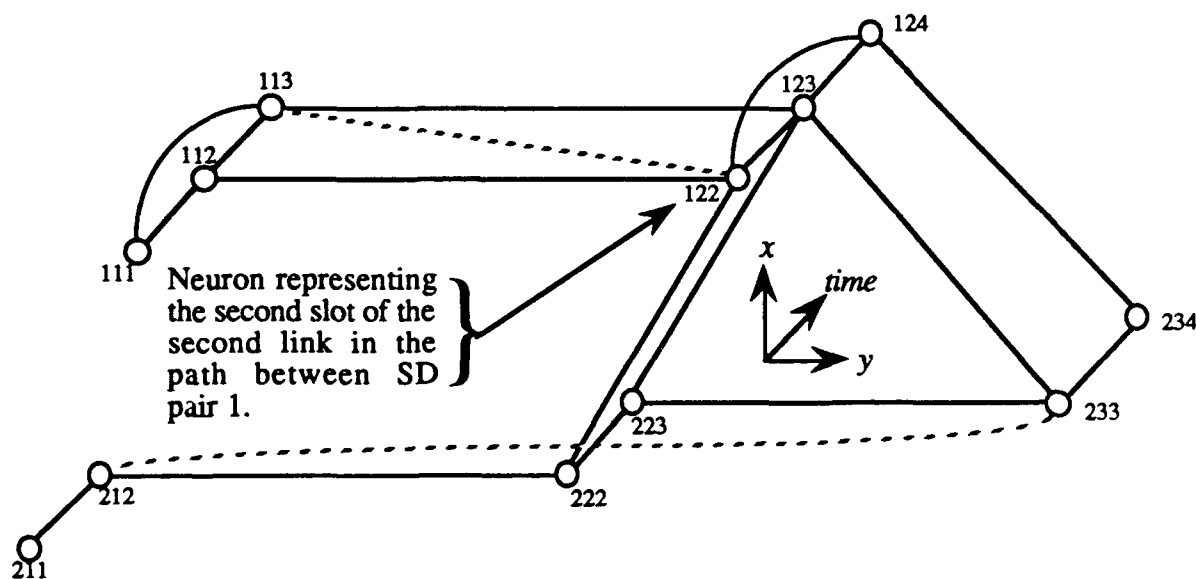


Figure 4.3. The reduced SAS model for the network shown in Fig. 4.1(a)

Table 4.1. An optimal schedule for the network of Fig. 4.1(a)
(X = link activation, o = open for activation, b = link blockage,
i = ineligible for activation owing to the blockage of an adjacent link)

Path	Link	Indices (SD, link)	Slot			
			1	2	3	4
1	(1->2)	1,1	X	b	i	
	(2->6)	1,2		X	b	b
2	(4->5)	2,1	X	o		
	(5->6)	2,2		b	X	
	(6->3)	2,3			b	X

4.5.3 The Adjustable-Length Model

Our approach with the basic, the condensed NAS, and the reduced SAS NN models, like the approach in [7], has been to attempt to solve the problem of determining the minimum-length admissible schedule that satisfies a given set of end-to-end communication requirements by repeatedly posing the binary question “Can a schedule be found that satisfies the given end-to-end demand in Λ slots?” for different values of Λ . Starting with Λ equal to a known lower bound on the schedule length, the question is repeated as Λ is incremented until an admissible schedule is found. Since the NN model only guarantees convergence to local minima of the energy function, the failure of any NN simulation to find a Λ -slot admissible schedule does not preclude the existence of such a schedule. Therefore multiple NN runs from different initial conditions must be run for a value of Λ that is too small before it may be concluded with any degree of confidence that a larger value is required. In the NAS problem a schedule can usually be found for the first value of Λ because the NAS bound B_{nas} is tight for many networks and communication requirements (the bound is tight for all the topologies we have examined). However, in the SAS problem multiple runs for several values of Λ are usually required because the SAS bound B_{sas} is generally not tight. These considerations, coupled with the frustration of multiple inadmissible solutions, led to the development of the “adjustable-length” model.

The adjustable-length model attempts to solve the scheduling problem without resorting to the binary-question approach. Ideally, every run of the adjustable-length NN model will deliver a conflict-free schedule of short, but not necessarily optimum, length. This is achieved by implementing the basic model (or the condensed NAS, or the reduced SAS variants) with an obviously excessive number of slots (e.g., use $\Lambda = 1.5 B_d$,⁸ a known upper bound on the minimum schedule length for NAS [6]), and penalizing activations that occur in later slots by means of an additional energy equation term, which we shall denote E_5 . Thus, link activations are encouraged in the early slots, and unused slots are discarded to yield a nearly minimum, if not minimum, length admissible schedule.

For problem instances in which the bound B_s (i.e., B_{nas} or B_{sas} , whichever is appropriate for the problem instance) is tight, the additional energy equation term can be simply formulated as an equality constraint:

$$E_5 = \sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{k=1}^{\Lambda} W(k) V_{ijk} = 0, \quad (8)$$

⁸ The bound B_d is the maximum nodal degree in the network, as discussed in Section 3.1.1.

where $W(k)$ can be defined as

$$W(k) = \begin{cases} (k - B_s)^2, & k > B_s \\ 0, & \text{otherwise} \end{cases}$$

or as some other similar function of k , the slot number. With this formulation, the penalty for activating links in slot k increases as k increases past B_s . However, as discussed in Section 3.1, it is generally not known a priori whether the bound is tight, and there are problem instances in which $E_5 > 0$ for all solutions that satisfy the first four constraints. Thus, in general, Eq. (8) should be written $E_5 \geq 0$.

Despite the fact that Eq. (8) should really be an inequality, we have found it advantageous to implement it as an equality constraint in our simulation runs. Doing so permits the use of a dynamically-varying Lagrange multiplier λ_5 , which continues to grow as long as the equality constraint is not satisfied. To prevent excessive growth of λ_5 when the bound B_s is not tight, a small value (relative to the other LM time constant values) is used for the time constant $(\Delta t)\lambda_5$. This causes the growth of λ_5 to be slower than that of the other LMs, so that violation of the constraint is, in essence, discouraged rather than prohibited. The resulting equation of motion term, which is found by applying Eq. (7), is simply

$$\frac{-\partial E_5}{\partial V_{ijk}} = -W(k).$$

4.5.4 Gaussian Simulated Annealing

Simulated annealing (SA) [23, 24, 25] is a probabilistic minimization algorithm that facilitates the escape from local minima so that the chances of finding the global minimum are enhanced. Under this technique, the energy function normally follows a gradient descent; however, random perturbations are applied to permit occasional transitions to states with higher energy. If these perturbations are large enough, it is possible to escape the local minimum, thereby permitting the search by gradient descent to resume in a new location in the search space.

In [15], we used a specialized form of SA, called Gaussian simulated annealing (GSA), which incorporates the use of a modified form of the "Gaussian machine" developed by Akiyama et al. [26, 27] in conjunction with the use of the method of Lagrange multipliers in some of our NN simulations. With GSA, the random perturbations are applied in the form of additive Gaussian noise (AGN), where the variance of the Gaussian noise diminishes with time.

The Gaussian Machine of [26, 27] used constant valued connection weights and combined a form of mean field annealing (MFA) [28] (an overview of MFA is given in Appendix A of [15]) with AGN. However, we have found that the combined use of the methods of Lagrange multipliers and AGN generally provides better results than those obtained through the use of MFA with AGN. Efforts to combine the use of MFA with the method of Lagrange multipliers led to the conclusion that there is no synergistic relationship between the two methods [15]. Therefore, our form of GSA employs the method of LM with AGN and an unchanging nonlinear neuron input/output voltage relationship.

The use of GSA has no direct effect on the NN energy formulation. Therefore, it may be used in conjunction with any of the NN models or their variants. The use of GSA is reflected in the equations of motion only by the additive noise term as follows:

$$u'_{ijk} = u_{ijk} + \eta,$$

where u_{ijk} is the input voltage in the absence of noise. This may also be written as

$$u'_{ijk}(t + \Delta t) = u'_{ijk}(t) - (\Delta t) \left(u'_{ijk}(t) - \sum_{l=1}^{N_{sd}} \sum_{m=1}^{L(l)} \sum_{n=1}^{\Lambda} T_{ijk,lmn} V_{lmn} - I_{ijk} \right) + \eta,$$

where u'_{ijk} is the input voltage in the presence of AGN, and the noise term η has a zero-mean Gaussian distribution with variance σ^2 . The variance is decreased according to a "cooling" schedule given by

$$\sigma = \frac{kT_o}{1 + t / \tau_T},$$

where $k = \sqrt{8/\pi}$, T_o is a parameter that controls the initial value (temperature) of the variance, and τ_T is the annealing time constant which controls the rate of cooling. After a specified number of iterations, G_{end} , η is set to zero so that noise is no longer added to the system. The use of this form of GSA has yielded improved performance in several of the scheduling NN models, as discussed in Sections 6 and 7.

5.0 SIMULATION ISSUES

Extensive simulation results have demonstrated the capability of our NN formulation to generate optimum or near-optimum schedules. Several variants of the NN model, which have used different versions of the constraints and/or Gaussian simulated annealing, have been

simulated. The results of these simulations are presented following a discussion of simulation methodologies and considerations.

The NN model was simulated using a program written in C++, which was run on Sun-4 workstations. The program reads a file that lists the nodes traversed in each of the predefined paths. This information is used to build the scheduling NN model, i.e., the neurons are defined and indexed such that neuron ijk represents slot k of the j^{th} link in the path between SD pair i (in the condensed NAS model, the neurons are double indexed so that neuron pk represents slot k of physical link p), the weight matrix T is created according to Eq. (6), and the bias currents are defined. The system parameters used in determining the coefficients in T and the bias currents are contained in a separate file to facilitate their modification as necessary. This process of "building" the NN is completely automated; the operator is only required to provide the file that enumerates the paths and, if desired, edit the parameter file to adjust the system parameters. Thus, different problem instances or networks may be quickly and easily analyzed.

The initial input voltage to each neuron ijk (pk for the condensed NAS model) is set so that

$$\sum_{k=1}^{\Lambda} V_{ijk}(0) = 1 \quad \left(\sum_{k=1}^{\Lambda} V_{pk}(0) = N_x(p) \text{ for the condensed NAS model} \right),$$

where $V_{ijk}(0)$ denotes the initial output voltage of neuron ijk . For the basic model, the output voltage is equal to the inverse of the number of slots Λ ; for the reduced SAS model the output voltage is equal to $(X_i + 1)^{-1}$ (recall that in the reduced SAS model, $X_i + 1$ neurons are created to represent the slots in which a link in the path between SD pair i may be legally activated, and that $X_i = \Lambda - L(i)$), and for the condensed NAS model the output voltage is equal to $N_x(p) / \Lambda$ (since physical link p should be activated in $N_x(p)$ slots). To obtain an initial output voltage of $V_{ijk}(0) = Y^{-1}$, we solve Eq. (1) for u_{ijk} , and find that the initial neuron input voltage must be set to

$$u_{ijk}(0) = u_o \tanh^{-1} \left(\frac{2}{Y} - 1 \right).$$

A small perturbation is then added to the initial input voltage of each neuron; addition of this perturbation avoids a totally symmetric initial state, which is an undesirable condition [10]. The perturbation quantity is randomly drawn from a uniform distribution on $[-0.1u_o, 0.1u_o]$, and is different for each neuron.

The equations of motion are iterated, allowing the NN to "relax" to a minimal energy state. It is important to note that system evolution is deterministic. The only randomness in the model is

associated with the choice of the initial state.⁹ To obtain different initial states for a given problem instance, different seeds are passed to a random number generator so that a unique sequence of random deviates is used to perturb the initial input voltages in each simulation. A particular initial condition can be reproduced by, once again, passing the random number generator the seed that produced the original state. Since the system evolution follows a trajectory of monotonically nonincreasing energy, the initial condition determines which portion of the solution space is actually searched, and thus which final state is reached.

5.1 A Binary Interpretation of the Analog State

A special feature of our problem is that each link must be activated in exactly one slot;¹⁰ i.e., exactly one of neurons ijk , $k = 1, \dots, \Lambda$ ($k = j, \dots, j + X_i$ for the reduced SAS model), must be activated. This property can be exploited by declaring the neuron with the largest output voltage in the set of neurons representing a link to be "on," regardless of its actual output voltage. Ties (i.e., equal output voltages) can be broken arbitrarily, e.g., by choosing the lowest-numbered neuron in such a set. Thus, at any time in the NN evolution, a tentative schedule may be obtained from the analog system state by picking a binary state in this manner. We refer to this state as the "instantaneous" state of the system. Tracking the instantaneous state permits the observation of the schedule as it evolves over time and seeks to eliminate scheduling conflicts. (The actual system evolution proceeds in the interior of the analog solution space, however. This mapping from an analog state to a binary one at each step of the iteration is simply for the purpose of assigning a binary interpretation to the state before termination has been reached.)

This instantaneous state interpretation may be easily applied to the condensed NAS model by simply declaring the $N_x(p)$ neurons with the largest output voltages in the set of neurons representing physical link p to be "on."

The instantaneous state interpretation of the analog state guarantees that constraints 2 (activate each link once and only once) and 3 (activate exactly N_x neurons) are satisfied. However, it also admits primary and sequence conflicts (violations of constraints 1 and 4, respectively). By assuming that those neurons that have been declared to be "on" have output voltages of 1, and all other neurons have output voltages of 0, we can count the number of primary and sequence conflicts by calculating the "binary-instantaneous" first and fourth constraint energies, respectively,

⁹ Except for the case of using simulated annealing in conjunction with our NN model, as discussed in Section 4.5.4, which does result in nondeterministic system evolution.

¹⁰ Except in the condensed NAS model where each physical link p must be activated in exactly $N_x(p)$ slots. This exception is covered in the next paragraph.

that result from the instantaneous state. Thus, the binary-instantaneous first constraint energy E_{h1} gives the number of primary conflicts, and the binary-instantaneous fourth constraint energy E_{h4} gives the number of sequence conflicts.

5.2 Termination Criteria

The iteration is terminated whenever one of the following four criteria is satisfied:

- (1) an admissible schedule has been discovered; i.e., when the instantaneous state is free of scheduling conflicts, or
- (2) a convergence has occurred; i.e., when all neuron output voltages are within some specified value ϵ (we have used $\epsilon = 0.01$ exclusively) of the output voltage limits 0 and 1,¹¹ or
- (3) a "stalemate" has been reached, i.e., when the instantaneous state has remained unchanged for a specified number N_c of iterations, or
- (4) a "time-out" is reached, i.e., when the NN has failed to find a legal schedule or otherwise terminate within a specified number N_{i-max} of iterations.

When the instantaneous state is free of scheduling conflicts, the corresponding binary-instantaneous constraint energy terms are zero. In particular, for the case of nonsequential-activation scheduling we have $E_{h1} = 0$; for the case of sequential-activation scheduling, $E_{h4} = E_{h1} = 0$. Since all of the constraints are essentially satisfied, continued iteration only serves to drive the output voltages of those neurons that are declared to be "on" (on the basis of the instantaneous state interpretation) nearer to their assumed output voltage value of 1, and force the output voltages of the remaining neurons toward 0. Thus, a conflict-free schedule of length Λ has been found and continued iteration can yield no improvements or new information.

In runs that are terminated because of convergence, stalemate, or time-out, the NN has failed to find a conflict-free schedule of length Λ . Termination due to convergence or stalemate generally indicates that the system is trapped in a high-energy local minimum of the energy function, and further iteration (without the aid of simulated annealing or some other mechanism to escape local minima) generally is futile. Termination as a result of a time-out may also indicate that the system is trapped in a high-energy local minimum of the energy function, or, more likely, that the system is waffling between two (or more) different inadmissible states. For example, an insufficient number of activations may provide sufficient excitation to activate neuron ijk , resulting

¹¹ Such convergence is always to an infeasible schedule; a feasible schedule would have been recognized prior to convergence by examining the instantaneous state.

in a primary conflict. This constraint violation in turn provides sufficient inhibition to deactivate neuron ijk , returning the system to the original condition of an insufficient number of activations.

5.3 Two Methods of Evaluating NN Performance

We have taken two different approaches, the *Monte-Carlo approach*, and the *multiple-instance approach*, in evaluating the performance of the NN models. With the Monte-Carlo approach, for a particular problem instance, a given value of Λ , and set of parameter values, the NN is run from a number of different initial states (typically 100). The fraction of runs that yield admissible schedules in the series of simulations is then used as a measure of the NN model's performance.

With the multiple-instance approach, a problem instance is simulated from different initial states until an admissible schedule is found. If an admissible schedule is not found within a specified number of different initial states N_{s-max} , Λ is incremented and the process is repeated. By applying this approach to a sequence of problem instances with similar characteristics, e.g., a sequence of problem instances that all have the same lower bound on schedule length, the performance of the NN model may be characterized by the average schedule length, and the average number of runs required to find a conflict-free schedule.

As discussed in Section 3.3, the 858 path sets that have $B_{nas} = 7$ were divided into two communication specification sets, set $(7, >7)$ and set $(7, 7)$. Set $(7, >7)$ consists of the 377 path sets that the NAS heuristic was unable to schedule in 7 slots, and set $(7, 7)$ consists of the 481 path sets that the NAS heuristic was able to schedule in 7 slots. In our simulation studies using the multiple-instance approach, we have compared these two sets in terms of the ability of the NN to generate minimum-length schedules.

5.4 A Modification that has Improved Simulation Results

We observed in our studies of a routing NN model [15] that an insufficient number of neurons were typically activated, a problem that was mitigated by increasing the bias currents. Hopfield and Tank [10] observed the same behavior in their studies of the TSP, as is discussed in Appendix A of [15]. In our studies of the scheduling NN model, insufficient neuron activation has not been a problem. Nonetheless, we have found that adjusting the neutral positions of the amplifiers with additional bias currents has helped in satisfying the system constraints. We have incorporated the additional bias current into constraint 3 (see Section 4.1), which may now be expressed as follows:

$$E'_3 = \frac{1}{2} \left(\sum_{i=1}^{N_{sd}} \sum_{j=1}^{L(i)} \sum_{k=1}^{\Lambda} V_{ijk} - \beta N_x \right)^2 = 0.$$

The typical value of β that was used in the routing problem was $\beta = 1.5$, which provided the additional excitation required to activate the correct number of neurons. This is also the value of β that was used by Hopfield and Tank [10] in their solution of the TSP. In the scheduling NN model, where the underactivation problem is minimal, we have found that setting $\beta \approx 0.61$ generally provides the best results. The resultant expression for bias currents, which should be compared to that given in Section 4.2, is:

$$I_{ijk} = \lambda_2 + \lambda_3 \beta N_x + I.$$

Methods to update the value of β dynamically, in conjunction with either the NAS or the SAS model, are discussed in Sections 6.2.3 and 6.4.1.

6.0 NAS SIMULATION RESULTS

In this section, we present the results of simulation of the somewhat-easier NAS problem in which the sequential-activation requirement is removed. In this case, the goal is to schedule each link the correct number of times in each activation cycle, without regard to the order in which the links of any path are activated. This approach supports the same total traffic (in terms of the number of activations of each link per cycle) as the sequential-activation model, but may result in greater end-to-end packet delay because several cycles may be needed to transport a packet from source to destination. Actually, in most cases, throughput (measured in terms of packets per slot) is greater using nonsequential-activation scheduling because removal of the sequentiality constraint often permits satisfaction of communication requirements in fewer slots.

In sections 6.1 - 6.3, the performance of the condensed NAS NN model is evaluated on the basis of Monte-Carlo simulations of three problem instances. The model that was found to give the best performance is the condensed NAS model with $\beta = 0.61$. The sensitivity of this model to parameter variations is also evaluated. In Section 6.4, heuristic improvements for the NN model are developed, and their effects are evaluated by means of multiple-instance simulations in Section 6.5. The results of simulations of the basic and the adjustable-length NAS NN models are presented in Section 6.6. Although, in general, the performance of these models is inferior to that of the condensed model with $\beta = 0.61$, each of the models has certain attributes that make its evaluation worthwhile.

6.1 The Condensed NAS Model Using the Monte-Carlo Approach

We have studied in detail the scheduling of the links corresponding to the communication requirements shown in Table 2.1 and Fig. 2.2 using different variants of the NAS model and the Monte-Carlo approach. As can be seen in Fig. 2.2, the maximum nodal degree in the network is 8 at node 13. Therefore, a lower bound on the nonsequential-activation schedule length for this problem is 8 slots. In fact, this is the minimum schedule length as verified by the existence of a number of admissible 8-slot schedules, which have been found by the NAS NN model.

We first considered the condensed NAS model with $\beta = 1$ and MLM; starting from 100 different initial states, an optimal schedule (conflict-free 8-slot schedule) was found in 84 runs. When GSA was used in conjunction with this model, simulations from the same 100 different states found 89 optimal schedules. However, the best results were obtained using the condensed NAS model with $\beta = 0.61$ and MLM. Optimal solutions were found using this value of β , both with and without GSA, in all of the simulations from 100 initial states. Out of the 200 optimal schedules found in these two simulations, no two were the same. This verifies that the NN is, in fact, searching different portions of the solution space and successfully converging to local minima of the energy function that correspond to optimal schedules. Despite the fact that both simulations used the same initial states, the application of GSA caused a completely different set of optimal schedules to be found. Thus, in this case, AGN alters the search trajectory without compromising (nor enhancing) the final solution quality. With GSA the average number of iterations required to find a schedule was 1075.7; in the absence of simulated annealing an average of 1112.7 iterations were required. The distribution of the number of iterations required to find a schedule is shown in Fig. 6.1. Both the minimum (255 iterations) and the maximum (8815 iterations) number of iterations required to find a schedule occurred when using GSA. The minimum and maximum simulation durations without GSA were 420 and 4375 iterations, respectively.

The parameter values used in the simulations of the condensed NAS models are shown in Table 6.1. In the table, $\lambda_c(0)$ denotes the initial value of all of the Lagrange multipliers λ_c 's (this includes the initial value of each of the MLM λ_{2ij} 's), and the parameter ζ is the limiting value of the neuron input voltages, i.e., $-\zeta \leq u_{pk} \leq \zeta$.¹² The GSA parameters apply only to the runs that used simulated annealing.

¹² Our studies have shown that it is helpful to place limits on the neuron input voltages (which otherwise could range from $-\infty$ to ∞); such limits help to prevent the neurons from being irrevocably locked into an "on" or "off" state.

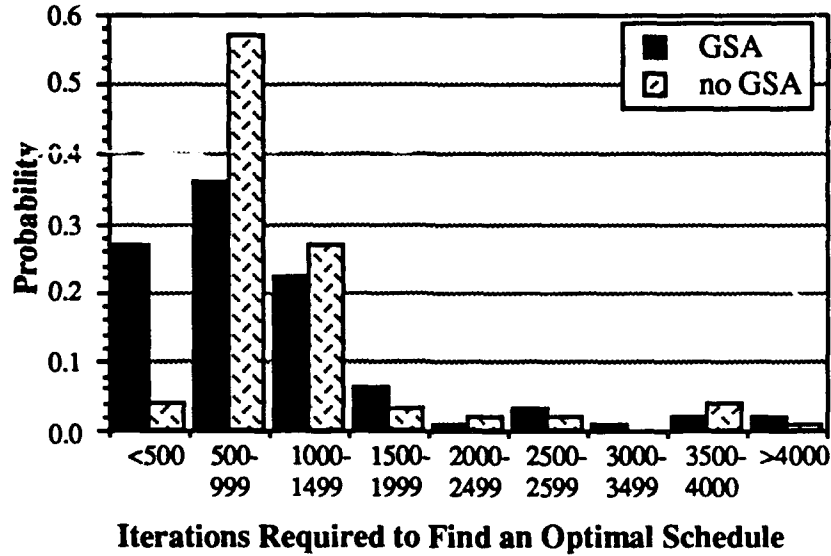


Figure 6.1. Distribution of the duration of simulations using the condensed NAS NN model with $\beta = 0.61$

Table 6.1. Condensed NAS NN parameters

$\lambda_c(0)$	$(\Delta t)_{\lambda_1}$	$(\Delta t)_{\lambda_2}$	$(\Delta t)_{\lambda_3}$	Δt	I	β	N_{i-max}	N_c	ζ	Λ	MLM	T_o	τ_T	G_{end}
1.0	0.02	0.1	0.1	10^{-4}	10	1.0, 0.61	2×10^4	10^4	0.75	8	yes	0.01	50	10^4

6.2 Parameter Sensitivity

An important issue in measuring the performance of a NN model is its parameter sensitivity. If *good* solutions are obtained using a wide range of parameter values, the time required to determine appropriate parameter values by multiple trial-and-error simulations is greatly reduced. It is also likely that, once an appropriate set of parameter values has been determined, this set of parameters will provide good NN performance over a broad class of problem instances. Analysis of parameter sensitivity also indicates which of the parameters cause the most variance in NN performance. Knowledge of these parameters gives a starting point for adjusting the parameter values for different communication requirements.

The condensed NAS model with $\beta = 0.61$ and the parameter values listed in Table 6.1 provided extremely good results when scheduling the requirements of Table 2.1. The sensitivity of this model to variations in the values of the bias parameters I and β , and the third constraint LM time constant $(\Delta t)_{\lambda_3}$ has been evaluated through simulations. It was found that the model is relatively insensitive to variations in any of these parameters. The most critical parameter is β , which, if set too small, prevents the discovery of admissible schedules by causing an oscillatory NN state. The results of these parameter sensitivity simulations are discussed next.

6.2.1 Bias Sensitivity

A series of Monte-Carlo simulations of the condensed NAS model with $\beta = 1.0$ was performed using different values for the additional bias I in each of the sets of runs from 100 different initial states. The communication requirements were given by Table 2.1 and Fig. 2.2. The parameter values of Table 6.1, with the exception of the bias values, were used in conjunction with GSA. The GSA parameter values are also listed in Table 6.1. The results of these simulations are shown in Fig. 6.2. The figure shows that over a bias range of 20, $[10, -10]$, the fraction of optimal schedules varied from a low of 89% to as high as 93%. The primary conclusions that may be drawn from these simulations is that the NN model is relatively insensitive to the bias value.

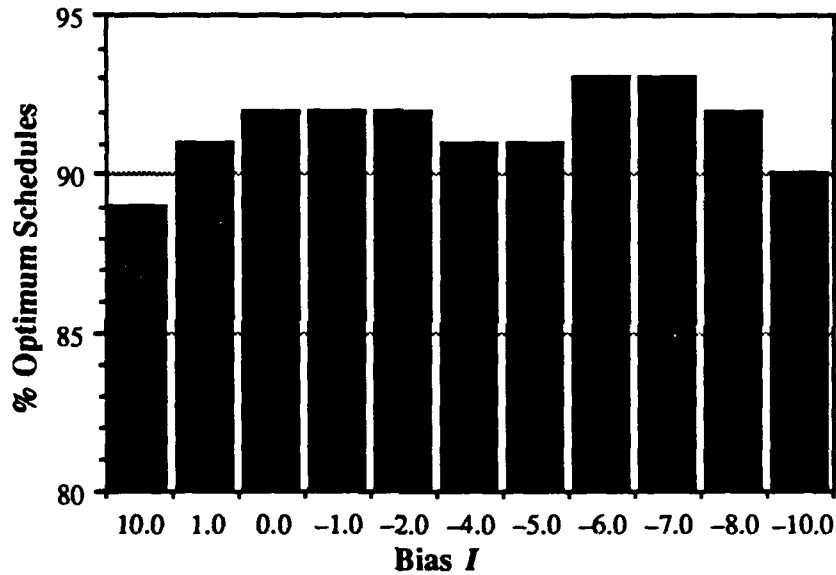


Figure 6.2. Success rate of the condensed NAS model using different values of I

6.2.2 LM Time Constant Sensitivity

A similar series of Monte-Carlo simulations of the condensed NAS model with $\beta = 1.0$ and MLM, both with and without GSA, were run using different values for the Lagrange multiplier λ_3 time constant $(\Delta t)_{\lambda_3}$, which corresponds to constraint 3. These simulations used the parameter values listed in Table 6.2 in efforts to schedule the communication requirements of Table 2.1. This series of runs was motivated by the remarkable performance of a distributed implementation of the link-neuron NN model for routing to minimize congestion [15]. In that model, the constraint that corresponds to λ_3 in the NAS model was "turned off." The resulting performance rivaled that obtained with any of the link-neuron routing models. If the same results are possible with the condensed NAS model, improved performance should be obtained as the third constraint time

constant $(\Delta t)_{\lambda_3}$ is decreased. However, as shown in Fig. 6.3, this is not the case. The figure shows the percentage of optimal solutions obtained using different values of $(\Delta t)_{\lambda_3}$, both with and without GSA. In the runs with $(\Delta t)_{\lambda_3} = 0$, the corresponding LM λ_3 was also set to zero so that the third constraint made no contribution to the energy E_{total} , nor to the equations of motion. Clearly, when using GSA, better performance is obtained when the system evolution is partially guided by the third constraint. However, the best performance was obtained in the absence of both GSA and the effects of the third constraint.

Table 6.2. Condensed NAS NN parameters for $(\Delta t)_{\lambda_3}$ analysis

$\lambda_c(0)$	$(\Delta t)_{\lambda_1}$	$(\Delta t)_{\lambda_2}$	$(\Delta t)_{\lambda_3}$	Δt	I	β	N_{i-max}	N_c	ζ	Λ	MLM	T_o	τ_T	G_{end}
1	0.02	0.01	vary	10^{-4}	0	1.0	2×10^4	10^4	0.75	8	yes	0.01	50	10^4

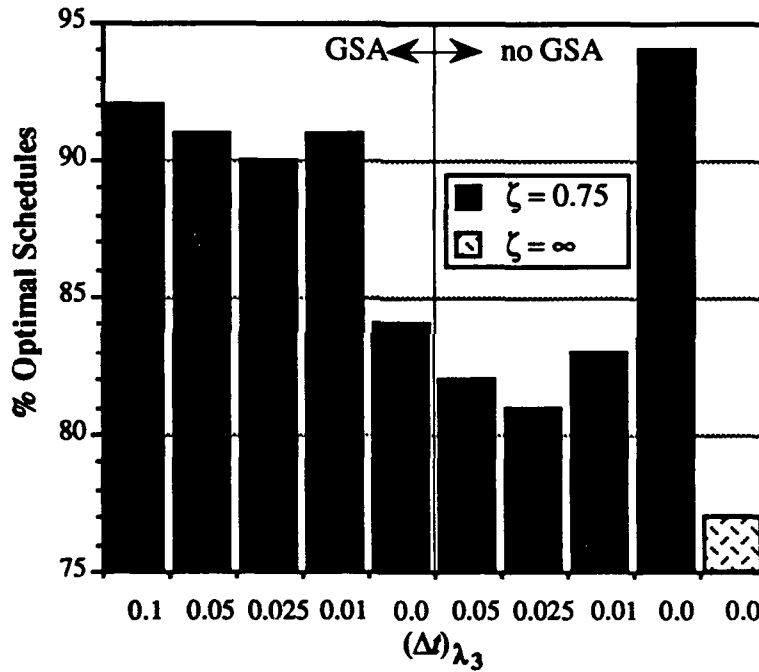


Figure 6.3. NAS NN success rate versus $(\Delta t)_{\lambda_3}$

Figure 6.3 also shows the benefit obtained by limiting the neuron input voltage to the interval $[-\zeta, \zeta]$. As indicated by the shading of the bars in the figure, in all but one of the simulations the input voltage to all of the neurons was restricted to $-0.75 \leq u_{pk} \leq 0.75$. The lighter shaded bar shows the results of a Monte-Carlo simulation in which the neuron input voltage values were not limited, but was otherwise a repeat of the Monte-Carlo simulation that yielded the best results, i.e., the simulation of the model that uses neither the third constraint, nor GSA. Clearly, limiting the input voltage results in a significantly increased number of optimal schedules. Limiting the input voltage allows the NN to respond more easily to forces exerted by the constraints by keeping the individual neuron states near the steep portion of the input/output nonlinearity.

6.2.3 Sensitivity to β

To determine the sensitivity of the NN model to the value of β , a series of simulations using MLM, GSA, the parameters of Table 6.1, and different β values were run in efforts to schedule the requirements of Table 2.1 without the sequentiality requirement. The nine different values of β that were used are shown in Fig. 6.4. For each value of β , the NN model was run from 50 different initial states. The same 50 initial states were used for each value of β .

We also tried varying β as a function of the first constraint energy E_1 . That is, we tried $\beta = f(E_1)$, where

$$f(E_1) = \begin{cases} 0.6, & \frac{E_1}{N_x} > r_t \\ 1 - \frac{E_1}{0.6N_x}, & \frac{E_1}{N_x} \leq r_t \end{cases} \quad (9)$$

Thus, the minimum value of β as a function of E_1 is 0.6, and β approaches 1.0 as E_1 approaches zero. The problem-specific parameter r_t sets the value of E_1 at which β begins its approach towards 1.0. In our current example, where $N_x = 41$, we set $r_t = 0.24$ so that as the value of E_1 drops below 10, β starts its monotonic growth towards 1.0. This approach was motivated by the fact that, if the iteration is allowed to continue after discovering a conflict-free instantaneous state, or if the NN fails to find an admissible schedule relatively early in the iteration, the LM λ_3 will eventually enforce the third constraint. With a nonunit value of β , a literal enforcement of the third constraint E_3' must result in ultimate convergence to an inadmissible solution (even after an optimal schedule has been found) because the wrong number of neurons ($\beta N_x \neq N_x$) must be activated. By allowing β to go to one as E_1 goes to zero, it was hoped that the benefit of the use of a nonunit value of β could be obtained without the associated degeneracies.

The results of these simulations are presented in Figs. 6.4 and 6.5. Figure 6.4 shows the probability of finding an optimal schedule using the different values of β . The figure indicates that the use of any value of β in the range $[0.488, 0.854]$ yields 100% optimal solutions to this problem. When values less than or equal to 0.244 were used, no admissible schedules were found. In fact, these small values resulted in an oscillatory state in which all neurons alternately had output voltages of 1.0 and then 0.0. The use of $\beta = f(E_1)$ yielded 90% optimal schedules, which was less than any of the runs with $0.366 \leq \text{constant } \beta \leq 1.0$. A different, and more successful, approach to the use of a nonconstant value of β is discussed in Section 6.4.1.

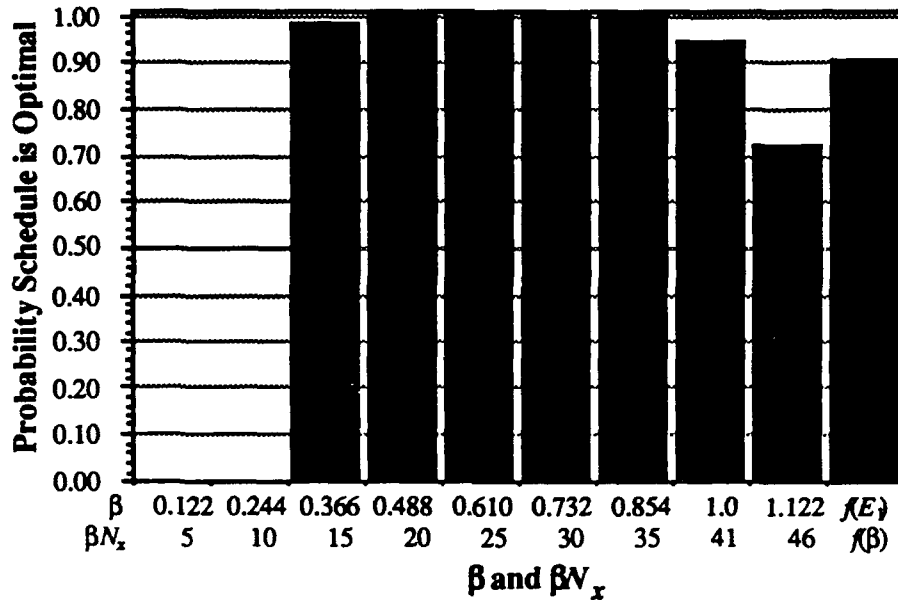


Figure 6.4. Probability of obtaining an optimal schedule versus β ($f(E_1)$ is given by Eq. (9))

Fig. 6.5 examines performance sensitivity with respect to β from a different viewpoint. In this figure, the average number of iterations required to terminate a run are plotted against the different values of β . The dark bars give the expected number of iterations for termination due to any of the termination criteria including time-out ($E\{\text{number of iterations}\}$). The light bars give the expected number of iterations to find an admissible schedule given that an admissible schedule is found ($E\{\text{number of iterations} \mid \text{optimal}\}$). Because most of the runs that failed to find an admissible schedule were terminated as a result of a time-out at $N_{i-\max} = 20,000$ iterations, we conclude that $E\{\text{number of iterations}\} \geq E\{\text{number of iterations} \mid \text{optimal}\}$, with equality only if 100% admissible solutions are found. The results for the runs with $\beta = 0.122$ and $\beta = 0.244$ are not shown because no admissible schedules were found in those runs, which all timed-out at $N_{i-\max} = 20,000$ iterations.

Figure 6.5 shows that the fewest number of iterations are required when $\beta = 0.61$ is used. This value yields $\beta N_x = 25$, which, perhaps coincidentally, is near to the number of physical links in the network (30). This raises a question as to whether $\beta = 0.61$ is a universally *good* value, or merely a good value for this problem instance. Perhaps β should be proportional to the ratio of the number of physical links N_{pl} to the number of required transmissions N_x , or it may be necessary to independently determine a good value of β by trial and error for every problem instance.

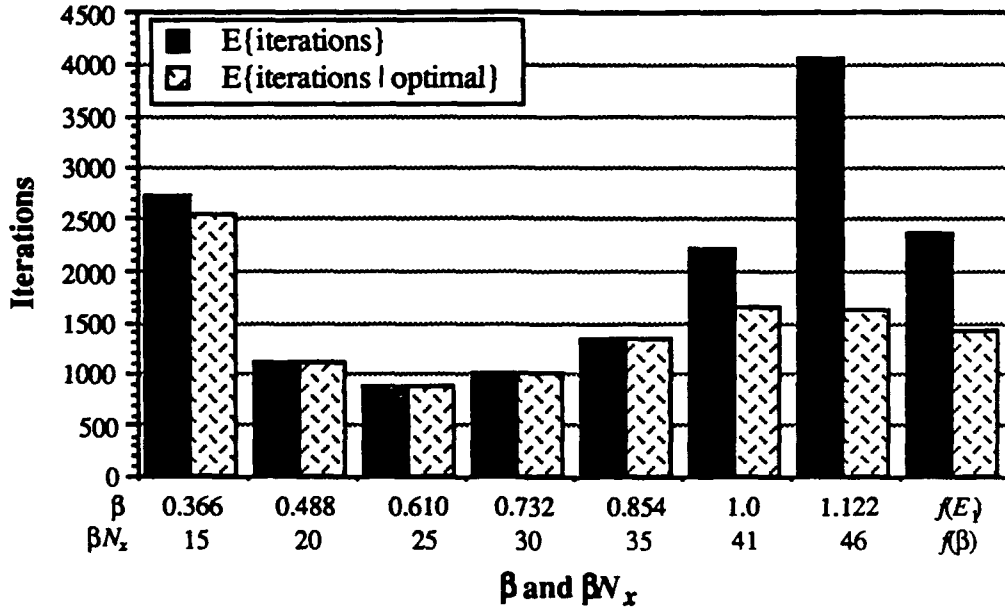


Figure 6.5. Expected number of iterations required to find an 8-slot schedule as a function of β ($f(E_1)$ is given by Eq. (9))

6.3 A More Difficult NAS Problem Instance

In an effort to address these concerns, Monte-Carlo simulations of a more difficult problem instance were run using the condensed NAS model with $\beta = 0.61$, and with $\beta = N_{pl} / N_x$. The communication requirements of this problem instance (referred to as the “augmented problem” in this subsection), which are listed in Table A2 of Appendix A, were generated simply by increasing the load on several of the physical links of Fig. 2.2. Here we refer to the problem instance defined by the communication requirements of Fig. 2.2 as the “original problem.” In this example, the transformation from the original to the augmented problem results in an increase in the value of N_x from 41 to 63, while the number of physical links in the network remains constant at 30. Thus $N_{pl} / N_x = 0.476$. The minimum schedule length for this problem is 8 slots. The NAS heuristic scheduled the problem in 9 slots.

Two series of simulations of the augmented problem were run from 100 different initial states using the condensed NAS model with MLM, GSA, the parameter values shown in Table 6.1, and the two different values of β . The simulations using $\beta = 0.61$ found 21 optimum schedules, whereas the simulations using $\beta = 0.476$ found only 8 optimal solutions. These results suggest that the use of a value of $\beta = 0.61$ might universally yield better performance than setting β proportional to a ratio of the number of physical links to the total number of transmissions required to satisfy the specification.

An example of one of the schedules found by the NN is shown in Table 6.3. In the "Link" column, the physical links are sequentially numbered and the adjacent nodes listed (see Fig. 2.1); e.g., the first entry in the link column, 1 (4,5), indicates that the physical link between nodes 4 and 5 has been labeled link 1. The " $N_x(p)$ " column lists the number of times the physical link must be activated. In the "Slot" columns an "X" indicates a physical link activation, and a "b" indicates that the physical link is blocked in that slot by some other activation. For example in slot 1, link 1 (4,5) is blocked by the activation of link 2 (5,13) and link 27 (2,4). Each of the cells in the "Slot" columns corresponds to a neuron; thus, an "X" entry corresponds to an instantaneous state neuron output voltage of 1. All of the remaining cells, both those with no entry and those with a "b" entry, correspond to neurons whose instantaneous state output voltages are 0. Note that in this schedule all but one slot are maximally scheduled; only one more legal link activation is possible in the slot that is not maximally scheduled, i.e., link 20 may be activated without conflict in slot 1.

A schedule in which there are no open slots, i.e., a schedule in which every neuron is either activated or blocked, is said to be a "maximal schedule." A schedule in which $y\%$ of the slots are either activated or blocked is referred to as a $y\%$ maximal schedule. Thus, the schedule shown in Table 6.3 is a 99.6% maximal schedule. Because the objective of the NN model is to find a link activation schedule that satisfies the specified communication requirements in a minimum number of slots, the percentage of slots that are blocked or activated does not give directly a measure of the quality of the schedule. Rather, it is indirectly related to optimality and gives an indication of the degree of difficulty involved in determining such a schedule. An admissible schedule of minimum length is optimal, regardless of how near the schedule is to being maximal.

It is hard to quantify precisely the difficulty of the scheduling problem for any particular problem instance, especially before a schedule has been generated. For example, although the minimum schedule lengths for both the original and the augmented problems discussed here are 8 slots, the augmented problem clearly has fewer degrees of freedom because 22 additional link activations are scheduled in the same time interval. Whereas only one more link activation is possible for the particular solution shown in Table 6.3, approximately 22 additional link activations are possible for typical schedules produced for the original problem. The ability of our NN model to produce a significant number of optimal solutions to such a highly-constrained problem illustrates the power of our method.

Table 6.3. An optimum (8-slot) schedule that satisfies the requirements of Table A2
(X = physical link activation, b = blocked slot)

Link	$N_x(p)$	Slot							
		1	2	3	4	5	6	7	8
1 (4,5)	2	b	b	b	X	b	b	X	b
2 (5,13)	3	X	b	X	b	X	b	b	b
3 (13,20)	2	b	b	b	b	b	X	b	X
4 (20,24)	1	b	b	b	b	b	b	X	b
5 (7,14)	3	b	b	b	X	b	X	X	b
6 (14,15)	3	b	X	X	b	X	b	b	b
7 (15,17)	1	b	b	b	b	b	b	X	b
8 (9,12)	1	b	b	b	X	b	b	b	b
9 (12,13)	1	b	b	b	b	b	b	X	b
10 (13,19)	2	b	X	b	X	b	b	b	b
11 (19,14)	2	X	b	b	b	b	b	b	X
12 (15,16)	2	b	b	b	b	b	X	b	X
13 (1,4)	1	b	X	b	b	b	b	b	b
14 (5,6)	3	b	X	b	b	b	X	b	X
15 (6,11)	2	b	b	b	b	X	b	X	b
16 (21,22)	3	b	b	b	b	b	X	X	X
17 (22,20)	5	X	X	X	X	X	b	b	b
18 (1,2)	3	b	b	b	X	b	X	X	b
19 (2,3)	3	b	X	X	b	X	b	b	b
20 (3,6)	1		b	b	X	b	b	b	b
21 (6,8)	1	b	b	X	b	b	b	b	b
22 (8,9)	4	X	X	b	b	b	X	b	X
23 (9,10)	3	b	b	X	b	X	b	X	b
24 (3,4)	1	b	b	b	b	b	X	b	b
25 (4,7)	2	b	b	X	b	X	b	b	b
26 (15,18)	2	X	b	b	X	b	b	b	b
27 (2,4)	2	X	b	b	b	b	b	b	X
28 (7,12)	1	b	X	b	b	b	b	b	b
29 (7,11)	2	X	b	b	b	b	b	b	X
30 (11,8)	1	b	b	b	X	b	b	b	b
unblocked		1	0	0	0	0	0	0	0

6.3.1 A Third NAS Problem Instance

Four combinations of parameters were used in examining a third problem instance, which is enumerated in Table A3 in Appendix A. The communication requirements of Table A3 result in a lower bound on the schedule length of $B_{nas} = 7$; however, the length of the schedule produced by the the NAS heuristic is 8 slots. Each of the four parameter combinations was used in attempts to schedule the problem in 7 slots from 100 different initial states (the same 100 initial states were used for each of the four parameter combinations). The parameter combinations used were $\beta = 1.0$ with GSA, $\beta = 1.0$ without GSA, $\beta = 0.61$ with GSA, and $\beta = 0.61$ without GSA. All of the

other parameter values were as listed in Table 6.1, except, of course, the number of slots Λ which was set to 7.

As shown in Fig. 6.6, the best results were obtained using $\beta = 0.61$: 67% of the runs resulted in admissible 7-slot (optimal) schedules when GSA was used. When GSA was not applied, 85% of the runs found optimal solutions. In contrast, the use of $\beta = 1.0$ in conjunction with GSA resulted in only 27% optimal schedules. The use of $\beta = 1.0$ without GSA resulted in 39% optimal schedules. In the runs with $\beta = 0.61$ that failed to find an optimal schedule, the conflicting constraints 2 and 3 that result from the use of a non-unit value of β caused the onset of an oscillatory NN state. This is not unexpected since, as discussed in Section 6.2.3, the failure to find an admissible schedule results in continued iteration of the equations of motion. With β set less than one, the Lagrange multiplier λ_3 grows in an effort to enforce the modified third constraint E_3' , which requires the activation of exactly βN_x neurons. Meanwhile, λ_2 attempts to enforce the second constraint, which requires exactly N_x neurons to be active. Thus, any success found by λ_3 results in violation of constraint 2 and increased growth of λ_2 , and conversely. As the conflict continues, the LMs, particularly λ_3 because of its impact on every neuron, become large enough to dominate the equations of motion. Then, when an insufficient number of neurons are active, an overly large λ_3 has an excitatory effect on every neuron, resulting in the activation of all neurons. At the next iteration, too many neurons are active, the effect of λ_3 is inhibitory, and all of the neurons are turned off. With the use of $\beta = 1.0$, no such oscillatory behavior was exhibited.

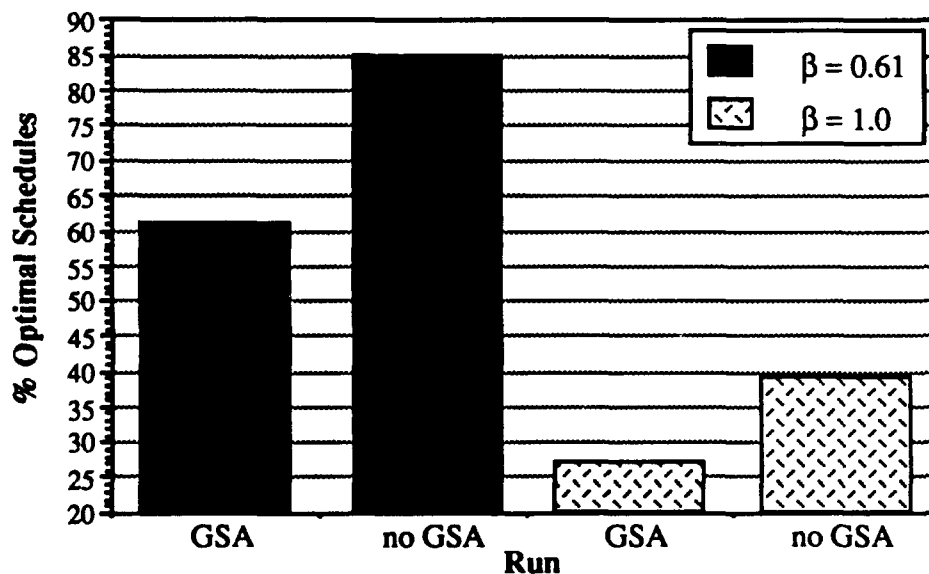


Figure 6.6. Results of efforts to schedule the requirements of Table A3 in 7 slots

This set of simulations with the requirements of Table A3, in contrast to the results of simulations with the requirements of Table 2.1, indicates that better results may be obtained in the

absence of GSA. In either case, however, the performance improvement as a result of the presence or absence of GSA was small in comparison to the improved performance as a result of the use of $\beta = 0.61$.

Although these simulations of three different problem instances using a few different β values constitute only a preliminary evaluation of the condensed NAS model with $\beta < 1$, they do indicate that the model is capable of finding minimum-length schedules in a large fraction of the runs. These simulations also indicate that the model is fairly robust to variations in the parameter values.

6.4 Some Improvements to the NN Model

6.4.1 Time-Varying β

The results of the Monte-Carlo simulations, which were presented above, have shown that improved NN performance is obtained by using the condensed model with β less than one. However, as discussed previously, the use of a constant, less-than-one value of β , presents two problems: First, continued iteration after the discovery of an admissible schedule must cause the NN state to change to an inadmissible schedule. Second, continued iteration as a result of the failure to discover an admissible schedule eventually leads to an oscillatory NN state. Therefore, a function that causes β to initially be approximately equal to 0.61, and to approach 1.0 as the NN converges, appears to be appropriate. In Section 6.2, the results of simulations indicated that using β equal to a function of the first constraint energy (Eq. (9)) was not satisfactory; 10% fewer optimal schedules were found using this function than were found using a constant value of β less than one. In this section, the use of two versions of a β -function that depends on the LMs λ_1 and λ_3 is explored via the multiple-instance approach.

The two versions of the function that have been considered are a monotonic nondecreasing β function, designated as β_m , and a nonmonotonic version of the same function, designated as β_n . These functions are updated at each iteration; thus at the $(n + 1)^{\text{st}}$ iteration, these equations are given by:

$$\beta_n(n+1) = \begin{cases} \beta', & \text{iterations} < 1000 \text{ OR } \lambda_3 < \lambda_1 \\ \beta' + \frac{(\lambda_3 - \lambda_1)(1 - \beta')}{2\lambda_1}, & \text{iterations} \geq 1000 \text{ AND } \lambda_1 \leq \lambda_3 < 3\lambda_1, \\ 1.0, & \text{otherwise} \end{cases}$$

and

$$\beta_m(n+1) = \begin{cases} \max\{\beta_m(n), \beta'\}, & \text{iterations} < 1000 \text{ OR } \lambda_3 < \lambda_1 \\ \max\left\{\beta_m(n), \beta' + \frac{(\lambda_3 - \lambda_1)(1 - \beta')}{2\lambda_1}\right\}, & \text{iterations} \geq 1000 \text{ AND } \lambda_1 \leq \lambda_3 < 3\lambda_1. \\ 1.0, & \text{otherwise} \end{cases}$$

Here β' is the minimum value that either of these functions is allowed to take, and has been set to $\beta' = 0.61$ in all of the simulations that have used one of these functions. This function, in either form, is intended to provide the benefits that have been obtained through the use of a small value of β . Through a minimum of 1000 iterations, or longer if λ_3 remains smaller than λ_1 , β maintains its minimum value. After 1000 iterations, the value of β is allowed to approach unity, thereby avoiding the onslaught of oscillations and allowing a tight convergence as an admissible state is reached. The growth of β is driven by the difference in the rate of growth between λ_3 and λ_1 . Recall that λ_1 is the LM associated with the first constraint (which prohibits primary conflicts), and λ_3 is the LM associated with the third constraint (which requires the activation of exactly N_x neurons). The value of λ_3 increases more rapidly than that of λ_1 not only because $(\Delta t)_{\lambda_3}$, the time constant associated with λ_3 , is five times as large as $(\Delta t)_{\lambda_1}$ (see Table 6.5), but also because a less-than-one value of β causes a conflict between the second and third constraints which, in turn, results in larger E_3 constraint-energy values.

6.4.2 A Traffic-Based Heuristic

Additionally, methods of extending conventional heuristic concepts to the NN model for nonsequential-activation scheduling have been examined. In the biased-greedy heuristic [3, 19], a bias, proportional to a node's degree, is applied to every node so that high-degree nodes are given a higher scheduling priority than lower-degree nodes. Thus, lower-degree nodes are scheduled in slots that are not blocked by the "prescheduled" high-degree nodes. This concept of scheduling high-degree nodes first may be extended to the condensed NAS NN model by applying a "traffic-based bias" to each physical link. This is accomplished either by setting the initial value of the MLM $\lambda_{2p}(0)$, or their associated time constants $(\Delta t)_{\lambda_{2p}}$, proportional to the degree of physical link p . The degree of link p , denoted by $\deg_\ell(p)$, is defined to be equal to the sum of the nodal degrees of the two nodes upon which link p is incident. For example, if link p is incident upon nodes r and r , $\deg_\ell(p) = \deg(r) + \deg(r)$. Setting $\lambda_{2p}(0)$ proportional to $\deg_\ell(p)$ results in a more stringent enforcement of the second constraint (which requires that physical link p be activated exactly $N_x(p)$ times) being applied to those neurons associated with higher-degree links in the early stages of the iteration. Setting $(\Delta t)_{\lambda_{2p}}$ proportional to $\deg_\ell(p)$ results in increased enforcement of the second constraint being applied to the neurons associated with high-degree nodes later in the iteration.

6.5 Evaluation of the NN Improvements via the Multiple-Instance Approach

The concepts developed in Section 6.4, i.e., the use of β_m , β_n , or a traffic-based bias to emphasize early scheduling of high-degree nodes (links), were evaluated by means of eight multiple-instance simulations of the condensed NAS model. In each of these simulations, the goal was to schedule the problem instances in set (7, >7) in 7 slots. These are the path sets that have $B_{nas} = 7$ but that the NAS heuristic was unable to schedule in 7 slots (see Table A5 in Appendix A for a partial listing). All eight of the simulations were able to find optimum schedules (i.e., admissible schedules of length 7) for each of the 377 problem instances in set (7, >7), thus verifying that the lower bound on all of these path sets is tight, and, more importantly, that the NN model is capable of delivering better schedules than the NAS heuristic.

The results of these simulations, which are labeled A through H, are shown in Fig. 6.7. The figure shows the average number of different initial NN states per problem instance required to find an optimal schedule. The parameter values that make each of the simulations unique are shown in Table 6.4, and the parameter values that were common to all of the simulations are listed in Table 6.5. In the following subsections, we discuss the effects of the improvements to the NN model that were described in Section 6.4.

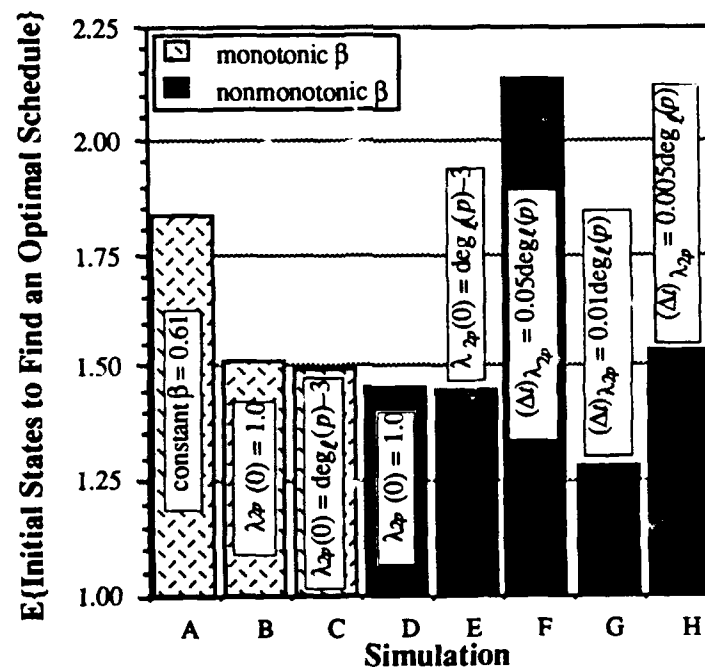


Figure 6.7. The average number of runs required to find an optimum schedule in simulations using variations of the condensed NAS model.

Table 6.4. Parameters used in multiple-instance runs of the condensed NAS NN model

Run	$\lambda_{2p}(0)$	$(\Delta t)\lambda_{2p}$	I	β
A	1	.01	10	0.61
B	1	.01	0	β_m
C	$\deg_{\ell}(p) - 3$.01	0	β_m
D, D'	1	.01	0	β_n
E	$\deg_{\ell}(p) - 3$.01	0	β_n
F	1	$.05\deg_{\ell}(p)$	0	β_n
G, G'	1	$.01\deg_{\ell}(p)$	0	β_n
H	1	$.005\deg_{\ell}(p)$	0	β_n

Table 6.5. Common parameters for Runs A - H

$\lambda_c(0)$	$(\Delta t)\lambda_1$	$(\Delta t)\lambda_3$	Δt	N_{i-max}	N_c	ζ	Λ	MLM	GSA
1	0.02	0.1	10^{-4}	11000	5000	0.75	7	yes	off

6.5.1 The Use of the Monotonic β -Function: Simulations A - C

Simulations B and C both employed the monotonic function β_m . Figure 6.7 shows that, as a result of the use of β_m , both simulations were more efficient in finding optimal schedules than simulation A, which was essentially a benchmark simulation that used the methods and parameter values (i.e., the condensed model with a constant β value of 0.61) that had been found effective in the Monte-Carlo simulations as described in Section 6.1. The fact that simulations B and C were more efficient than simulation A is noteworthy because the ability of simulation A to find an optimal schedule in less than two (1.83) different initial states, on average, appears to indicate good NN performance. In conjunction with the use of β_m , a traffic-based bias was applied by setting the initial values of the MLM $\lambda_{2p}(0)$ proportional to $\deg_{\ell}(p)$ in Simulation C. This yielded only a slight performance improvement over that of simulation B, in which no traffic-based bias was employed. Thus, the use of β_m is the primary cause of the significantly improved performance of simulations B and C, when compared to simulation A; i.e., the use of the monotonic function β_m delivers much better results than the use of a constant value for β .

6.5.2 The Use of the Nonmonotonic β -Function: Simulations D - E

Figure 6.7 shows that both simulations D and E delivered better performance than either simulations B or C. This is because the nonmonotonic function β_n was used in simulations D and E, as opposed to the monotonic function β_m which was used in simulations B and C. Otherwise, the NN models used in simulations B and D are identical, as are the models used in simulations C and E. As was the case in simulations B and C, the inclusion of a traffic-based bias ($\lambda_{2p}(0)$)

proportional to $\deg_{\ell}(p)$) caused the performance of simulation E to be slightly better than that of simulation D, in which no traffic-based bias was used. However, the principal conclusion to be drawn from these simulations is that the use of the nonmonotonic function β_n yields better performance than can be obtained through the use of the monotonic function β_m .

6.5.3 The Use of a Traffic-Based Bias: Simulations E – H

Each of the simulations E, F, G, and H employs some form of traffic-based bias in conjunction with the use of β_n . Simulation E used the form of traffic-based bias in which the initial values of the MLM $\lambda_{2p}(0)$ are proportional to $\deg_{\ell}(p)$. As shown in Fig. 6.7 and discussed in Section 6.5.2, the effects of this form of traffic-based bias are nearly negligible.

In simulations F, G, and H, separate time constants $(\Delta t)_{\lambda_{2p}}$ were introduced to correspond to each of the Lagrange multipliers of the form λ_{2p} . These time constants were assigned values proportional to the degrees of their associated links, with the constant of proportionality for each of the simulations listed in Table 6.4. Figure 6.7 shows that the value of the constant of proportionality greatly impacts the NN performance. With this constant set to 0.05 (simulation F), an average of 2.13 runs are required to find a schedule. This is the least-efficient performance of any of the multiple-instance simulations. However, with the constant of proportionality set to 0.01 (simulation G), the most efficient performance of any of the simulations was obtained. This simulation required an average of 1.28 different initial states to find a schedule.

The results of simulations E - H indicate that the use of a traffic-based bias can significantly improve the NN performance. Application of the bias to the LM time constants has the largest impact; however, this impact is beneficial only when the constant of proportionality is set to an appropriate value. Application of the bias to the initial value of Lagrange multipliers of the form λ_{2p} yields only slightly improved NN performance. The use of either form of traffic-based bias increases the NN complexity, and introduces new parameters whose best values must be determined by trial and error. However, the performance achieved in simulation G indicates that the benefits from the use of the traffic-based bias (applied to the LM time constants) outweigh the added NN complexity.

6.5.4 NN Scheduling of Set (7, 7)

To further verify the ability of our NN model to find optimum schedules, two multiple-instance simulations of the problem instances in set (7, 7) were run with the objective of finding 7-slot schedules for each of the path sets. Recall that set (7, 7) consists of the 481 path sets that the NAS heuristic was able to schedule in $B_{nas} = 7$ slots (see Table A6 in Appendix A for a partial

listing). The simulations, which are labeled D' and G', used the same sets of parameter values as were used in simulations D and G, respectively (see Tables 6.4 and 6.5 for the parameter values, and Sections 6.5.2 and 6.5.3 for discussions of the NN models). Both of these simulations were able to discover optimum schedules for each of the problem instances. The results of these simulations, in terms of the average number of different initial NN states required per problem instance, are compared with those from simulations D and G (which scheduled the problem instances in set (7, >7) in Fig. 6.8. A comparison of the results of simulations D' and G' to those of simulations D and G, respectively, indicates that the problem instances in set (7, >7) are, in fact, more difficult to schedule (on the average) than those in set (7, 7).

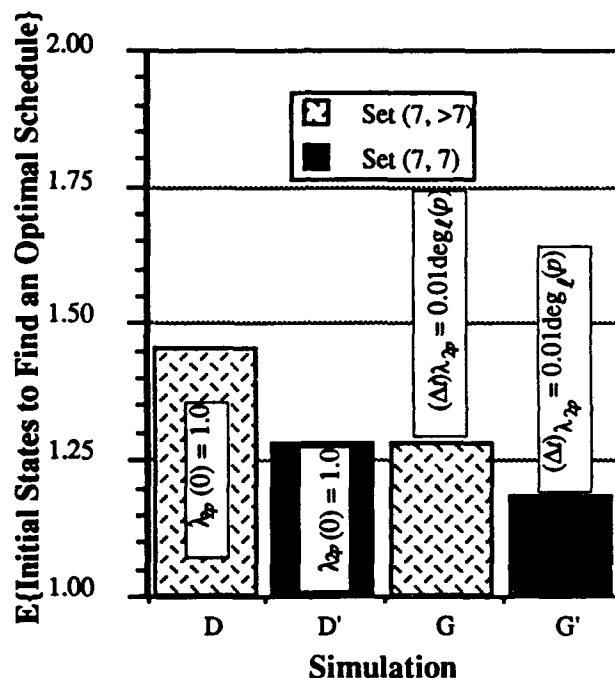


Figure 6.8. The average number of runs required to find optimum schedules for problem instances in set (7, >7) and set (7, 7)

6.6 An Evaluation of the Basic and the Adjustable-Length NAS Models

The basic and the adjustable-length NAS NN models offer certain advantages over the condensed NAS model and its variants, even though they do not provide comparable performance. The main advantage of the basic model is its simple formulation, which may be relatively easily implemented. The adjustable-length model, on the other hand, is a more complex formulation. It adds one constraint, and requires approximately 1.5 times the number of neurons required in the basic model. Nonetheless, the adjustable-length model potentially yields an admissible, if not optimal, schedule from every run. It also provides an important step toward the development of a joint routing-scheduling NN model, which is discussed in Section 8.

We have performed 12 Monte-Carlo simulations to evaluate the performance of the basic and the adjustable-length NAS models. The simulations are labeled A through F (no relationship to the simulations A through H that were presented in Section 6.5), and A' through F'. Simulations A through F used the basic model, and simulations A' through F' used the adjustable-length model, to schedule the communication requirements of Table 2.1. Recall that $B_{nas} = 8$ for this example; thus the minimum schedule length that can be found is 8 slots. The parameter values used in these simulations are shown in Table 6.6.

Table 6.6. NAS NN parameters used in simulations of the basic and the adjustable-length models

Run	$\lambda_c(0)$	$(\Delta t)_{\lambda_1}$	$(\Delta t)_{\lambda_2}$	$(\Delta t)_{\lambda_3}$	$(\Delta t)_{\lambda_5}$	Δt	I	N_{i-max}	N_c	ζ	T_o	τ_T	G_{end}
A, A'	1	0.01	0.1	0.1	0.01	10^{-4}	10	2×10^4	10^4	∞	off	off	off
B-F, B'-F'	1	0.02	0.01	0.01	0.01	10^{-4}	10	2×10^4	10^4	0.75	.01	50	10^4

The results of Runs A through F are shown in Fig. 6.9, and the results of Runs A' through F' are shown in Fig. 6.10. The similarities and differences between each of the runs are also summarized in the figures. The data shown in these two figures illustrates four important points:

Point 1: The use of $\beta = 0.61$, yields significantly better results than the use of a unit-valued β in all of the NAS models. This can be seen in Fig. 6.9, where, using $\beta = 0.61$, Run D found nearly twice as many optimal schedules as Run C, which used $\beta = 1.0$. The only difference between these two runs was the value of β . In the condensed basic model (Runs E and F, Fig. 6.9), the adjustable-length model (Runs C' and D', Fig. 6.10), and the condensed adjustable-length model (Runs E' and F', Fig. 6.10), the performance is also notably improved by using a less-than-one value of β , rather than a unit value of β , although the improvement is not as large as that seen with the basic model.

Point 2: The use of the condensed model in conjunction with the use of $\beta = 0.61$ further improves the performance of both the basic and the adjustable-length models. With $\beta = 0.61$ in Run F, the condensed basic model found 100% optimal schedules, which is 5% more than were found by its uncondensed counterpart in Run D (see Fig. 6.9). The condensed adjustable-length model used in Run F' found 7% more admissible and 4% more optimal schedules than the uncondensed model used in Run D' (see Fig. 6.10). With a unit β value, "condensing" the basic model (compare condensed Run E to uncondensed Run C in Fig. 6.9) also yields a much larger percentage of optimal solutions. However, as can be seen by comparing Run E' to Run C' in Fig. 6.10, "condensing" the adjustable-length model that uses a unit-value for β yields mixed results; it increases the percentage of admissible solutions while reducing the percentage of optimal solutions.

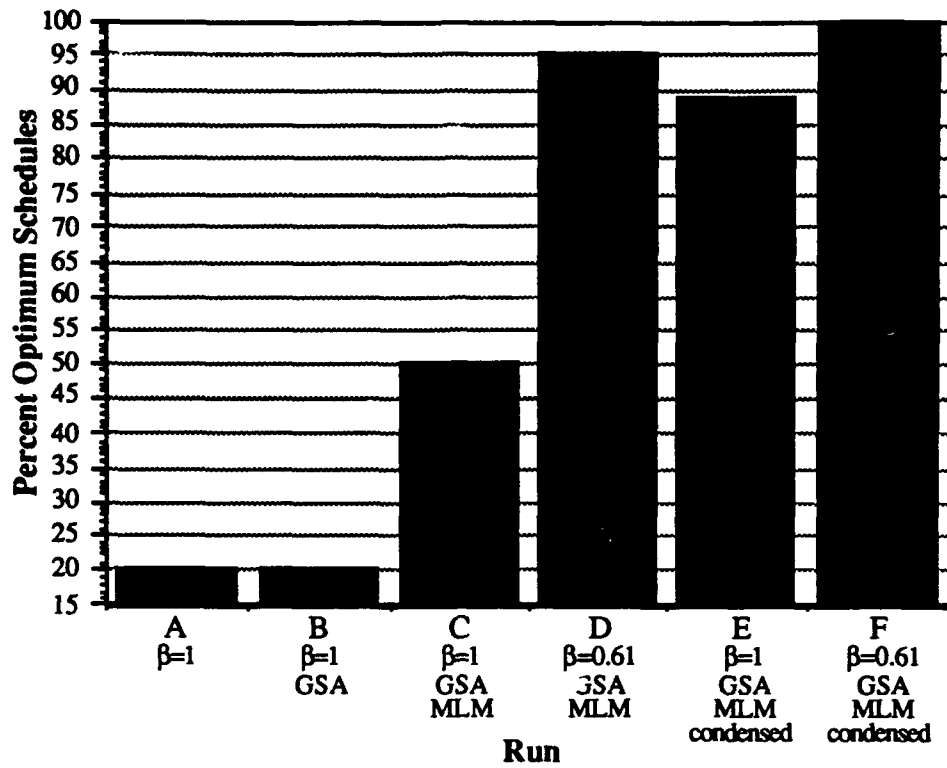


Figure 6.9. Results of simulations of variations of the basic NAS NN model ($\Lambda^* = 8$ slots)

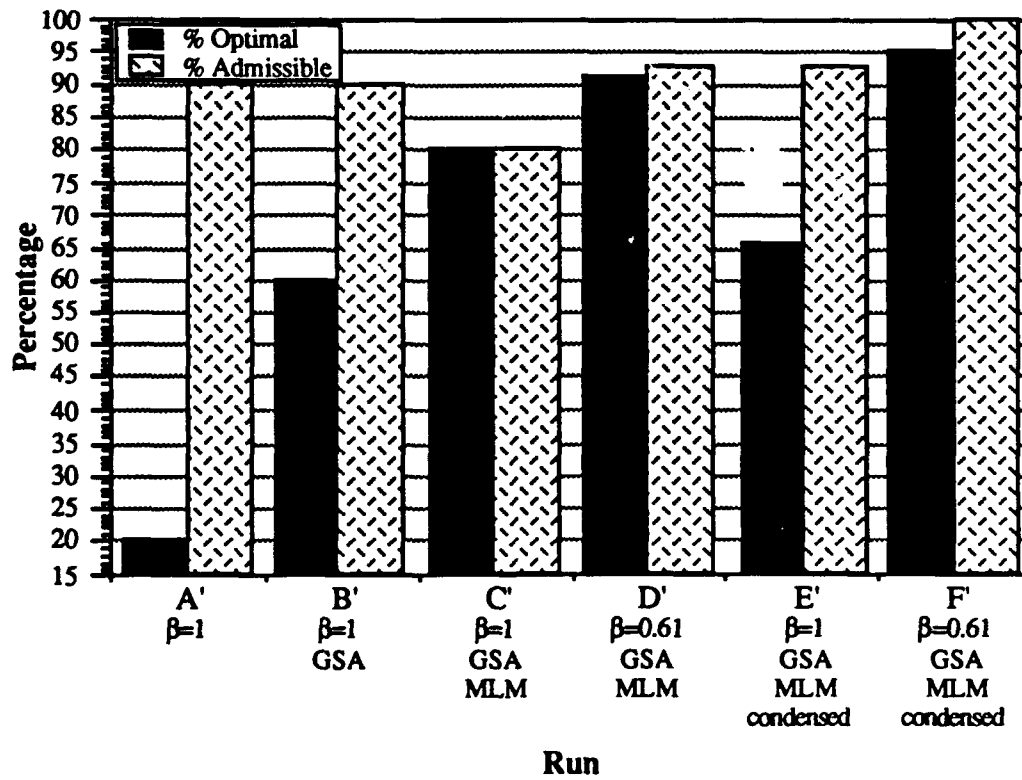


Figure 6.10. Results of simulations of the adjustable-length NAS NN model ($\Lambda^* = 8$ slots)

Point 3: Both GSA and MLM make important contributions to the NN performance. In the absence of either of these aids, neither the basic (Run A, Fig. 6.9) nor the adjustable-length (Run A', Fig. 6.10) models were able to find more than 20% optimal schedules. But with the use of GSA and MLM, the percentage of optimal schedules found by the basic model (Run C, Fig. 6.9) was increased to 50%, and the adjustable-length model (Run C', Fig. 6.10) found 80% optimal schedules.

Point 4: All variants of the adjustable-length model deliver a reasonably high percentage of admissible schedules. As discussed in Section 4.5.3, the purpose of the adjustable-length NN model is, primarily, to deliver admissible schedules consistently, and, secondarily, to yield nearly minimum, if not minimum, length schedules. As shown in Fig. 6.10, at least 90% of the schedules found in each simulation of the adjustable-length model, except Run C', are admissible. In Run C', 80% of the schedules are admissible. Thus, although this model does not yield an admissible schedule from every initial state, it does yield admissible schedules from a large fraction of starting points, regardless of the underlying model variant. Furthermore, simulations of both the condensed and the uncondensed adjustable-length models using $\beta = 0.61$, GSA, and MLM (Runs F' and D' respectively) found more than 90% optimal schedules, demonstrating that the adjustable-length model is also capable of delivering minimum-length schedules consistently.

6.7 Conclusions on the NAS NN Model

In this section, we have presented the results from simulations of three different NAS NN models, the basic, the condensed, and the adjustable-length models. In both Monte-Carlo and multiple-instance simulations, very satisfactory results were obtained using the condensed NAS model with $\beta = 0.61$ and MLM. In Monte-Carlo simulations of the problem instance given by Table 2.1, 100% of the schedules found by this model were optimal. Simulations of the heavily-congested problem instance given by Table A2, which required near-maximal scheduling, resulted in 21% optimal solutions, thus demonstrating the ability of our method to determine optimal schedules for highly-constrained problems. The model was also able to schedule optimally all of the problem instances in sets (7, 7) and (7, >7), thereby demonstrating the ability of the NN model to produce optimal schedules for many instances for which the NAS heuristic was unable to do so. The efficiency of the condensed NAS model was increased by the introduction of a traffic-based heuristic and the use of a time-varying β . Multiple-instance simulations of set (7, >7) with this more-efficient model found optimal solutions in an average of 30% fewer runs than were required by the condensed NN model that used $\beta = 0.61$ and MLM. These simulations of the condensed NAS NN model have shown that the model is capable of finding minimum-length schedules in a large fraction of the runs. They also indicate that the model is fairly robust to variations in the parameter values and in the communication requirements. Thus, we feel that the model is applicable to, and will perform well in, a broad class of scheduling problems.

The basic and the adjustable-length NAS NN models offer certain advantages over the condensed NAS model and its variants, even though they do not provide comparable performance. The basic model provides the foundation on which all of the other NAS NN models are built. The adjustable-length model (usually) provides an increased percentage of admissible schedules. In addition, because this model does not require an accurate estimate of the minimum schedule length, it provides an important step toward the development of a joint routing-scheduling NN model for which such an estimate may be difficult. Simulations of both the basic and the adjustable-length models have demonstrated their ability to deliver reasonably good performance.

7.0 SAS SIMULATION RESULTS

All of the SAS simulations have used some form of the reduced SAS model. The use of this model, besides reducing the dimensionality of the solution space by eliminating a number of inadmissible solutions, also significantly reduces the number of neurons in the NN. Since the number of computations required at each iteration of the NN model is approximately proportional to the square of the number of neurons, a reduction in the number of neurons markedly reduces the time required to complete a simulation.

7.1 Sequential-Activation Scheduling with the Monte-Carlo Approach

The minimum length of a sequential-activation schedule that satisfies the communication requirements of Table 2.1 with no scheduling conflicts is 9 slots. Table 7.1 lists one such schedule that was found by a SAS NN model in Run D,¹³ which will be discussed soon. In the table, entries are made only in the cells that correspond to a neuron in the reduced SAS model. The notation used is the same as that used in Table 4.1 in Section 4.5.2: a cell entry of "X" denotes a link activation, "o" denotes an open slot (i.e., activation of the link would not result in conflict), "b" denotes a blocked slot (i.e., one in which activation will result in a primary conflict), and "i" denotes an ineligible slot (i.e., a slot in which activation of the link must cause a sequence conflict). The blank cells represent the slots in which the link can never be activated in an admissible sequential-activation schedule. Of the 225 cells containing an entry (each of which represents a neuron), there are 34 "open" cells. Thus, the schedule shown in Table 7.1 is an 84.9% maximal schedule. In Section 7.5.4 we discuss the schedule of a more highly-constrained problem in which the choice of path sets permits satisfaction of the communication requirements in eight slots, resulting in a schedule that is 96.4% maximal.

¹³ In this section (Section 7), although the alphabetically labeled simulations may have the same labels as those in Section 6, they are unrelated.

Table 7.1. An admissible 9-slot schedule satisfying the requirements of Table 2.1.

(X = Link activation, o = open for activation, b = link blockage,
i = ineligible for activation owing to the blockage of an adjacent link)

Path†	Link	Indices (SD, link)	Slot								
			1	2	3	4	5	6	7	8	9
0	(4->5)	1,1	b	X	o	b	b	b			
	(5->13)	1,2		b	b	X	b	b	b		
	(13->20)	1,3			b	b	X	b	b	b	
	(20->24)	1,4				i	b	b	b	o	X
6	(7->14)	2,1	X	b	b	b	i	b	b		
	(14->15)	2,2		X	b	b	i	b	b	b	
	(15->17)	2,3			o	X	o	b	b	b	b
10	(9->12)	3,1	X	b	i	i					
	(12->13)	3,2		X	b	b	b				
	(13->19)	3,3			X	b	b	b			
	(19->14)	3,4				X	o	b	b		
	(14->15)	3,5					o	X	b	b	
	(15->16)	3,6						b	X	b	b
12	(1->4)	4,1	b	b	o	X	b	b			
	(4->5)	4,2		b	i	b	X	b	b		
	(5->13)	4,3			b	b	b	X	b	b	
	(13->19)	4,4				b	b	b	b	b	X
20	(5->6)	5,1	X	b	b	b	b	b	o	b	
	(6->11)	5,2		o	b	b	o	X	o	o	b
23	(21->22)	6,1	X	o	o	o	o				
	(22->20)	6,2		o	o	o	b	X			
	(20->13)	6,3			b	b	b	b	X		
	(13->5)	6,4				b	b	b	b	X	
	(5->6)	6,5					b	b	i	b	X
36	(1->2)	7,1	X	b	o	b					
	(2->3)	7,2		X	b	o	i				
	(3->6)	7,3			X	b	o	b			
	(6->8)	7,4				X	o	b	o		
	(8->9)	7,5					o	X	o	o	
	(9->10)	7,6						b	o	o	X
42	(3->4)	8,1	X	b	b	b	b				
	(4->7)	8,2		b	b	b	b	X			
	(7->14)	8,3			b	b	i	b	X		
	(14->15)	8,4				b	i	b	b	X	
	(15->18)	8,5					i	b	b	b	X
45	(2->4)	9,1	b	b	o	b	b	b	X		
	(4->7)	9,2		b	b	b	b	b	b	X	
	(7->12)	9,3			b	b	i	b	b	b	X
50	(14->7)	10,1	b	b	X	b	i	b	b		
	(7->11)	10,2		i	b	X	o	b	b	b	
	(11->8)	10,3			i	b	o	b	o	o	X

† See Table A1 in Appendix A.

In an effort to find 9-slot (optimal) sequential-activation schedules that satisfy the communication requirements of Table 2.1, a Monte-Carlo simulation, Run C, of the reduced SAS model with MLM and GSA was run from 100 different initial states using the parameter values shown in Tables 7.2 and 7.3. Previous efforts without MLM, GSA, or neuron input voltage limiting, as shown in Table 7.2, had found only 2% optimal schedules (Run B), and, in fact, were able to find 10-slot sequential-activation schedules in only 8% of the simulations of Run A. With the use of MLM, GSA, and input voltage limiting in Run C, the NN was able to find 9-slot schedules in 7% of the simulations. The distribution of the number and type of scheduling conflicts as a result of the schedules found by Run C are shown in Fig. 7.1.

Table 7.2. Summary of the results from, and differences between, Runs A - C, simulations of the reduced SAS NN model

Run	$(\Delta t)_{\lambda_1}$	$(\Delta t)_{\lambda_{2ij}}$ $(\Delta t)_{\lambda_3}$	Δt	I	N_{i-max}	ζ	Λ	GSA	MLM	% admissible	% optimal
A	0.01	0.05	10^{-5}	5	25000	∞	10	no	no	8	0
B	0.01	0.1	10^{-5}	5	25000	∞	9	no	no	2	2
C	0.02	0.1	10^{-4}	10	11000	0.75	9	yes	yes	7	7

Table 7.3. Common parameters for Runs A - C
(the GSA parameters apply only to Run C)

$\lambda_c(0)$	$(\Delta t)_{\lambda_4}$	N_c	T_o	τ_T	G_{end}	MLM
1	0.01	10^4	0.01	50	10^4	yes

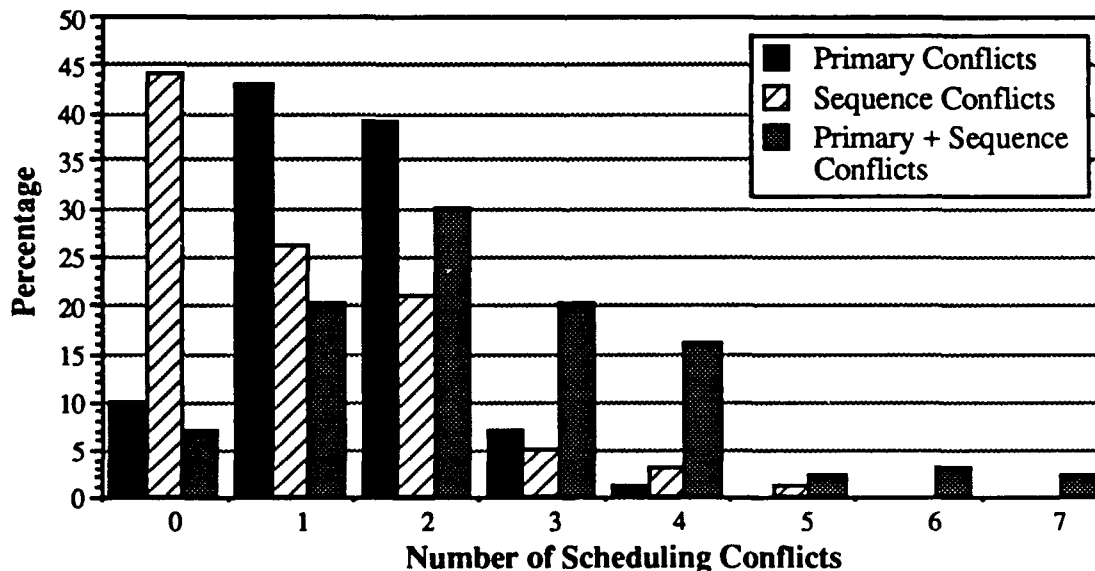


Figure 7.1. Results from Run C, a Monte-Carlo simulation of the reduced SAS model with GSA and MLM

In Fig. 7.1, the black bars represent primary conflicts, the striped bars represent sequence conflicts, and the gray bars represent the total number of scheduling conflicts, both primary and sequence. For example, the gray bar representing two scheduling conflicts includes the three cases of: no primary conflicts and two sequence conflicts, one of each type, or two primary conflicts and no sequence conflicts. Referring to the bars for zero conflicts, the figure shows that approximately 10% of the schedules found in Run C had no primary conflicts, 44% had no sequence conflicts, and 8% were optimal schedules because they had no conflicts of either type. The figure indicates that the NN model was more successful in enforcing the sequential-activation constraint than the primary conflicts constraint, even though the primary conflicts time constant $(\Delta t)_{\lambda_1}$ was twice as large as the sequence conflicts time constant $(\Delta t)_{\lambda_4}$ (see Tables 7.2 and 7.3).

7.2 Skewed Initialization

As a consequence of the disappointing results obtained in Run C, which are markedly inferior to the results obtained for the NAS model (without the heuristics that were subsequently developed for it), it has been necessary to develop heuristics to improve NN performance. As was done in modifying the NAS NN model, the concepts used in developing a heuristic SAS algorithm were applied to the SAS NN model. Much improved SAS performance was obtained by, in essence, setting the initial neuron output voltages so that the initial instantaneous state was free of sequence conflicts. This is done by setting the initial output voltage value of neuron ijk to

$$V_{ijk}(0) = \begin{cases} \frac{1}{X_i + 1} + m_i \left(k - j - \frac{X_i}{2} \right), & \frac{1}{X_i + 1} + m_i \left(k - j - \frac{X_i}{2} \right) > 0 \\ 0.5, & \frac{1}{X_i + 1} + m_i \left(k - j - \frac{X_i}{2} \right) \leq 0 \end{cases}, \quad (10)$$

where $X_i + 1 = \Lambda - L(i) + 1$ is the number of neurons that represent each link in the path connecting SD pair i , and m_i is the slope of the initial output voltage $V_{ijk}(0)$ as a function of the time slot k that neuron ijk represents. This function is plotted in Fig. 7.2. We call this type of neuron initialization a "skewed initialization." We have used $m_i < 0$; however, positive values of m_i can also be used, as discussed below. Setting $m_i = 0$ results in the original form of initialization, as discussed in Section 5.0. As in Section 5.0, a random perturbation is added to each neuron following skewed initialization of the NN so that different random seeds cause different portions of the solution space to be explored.

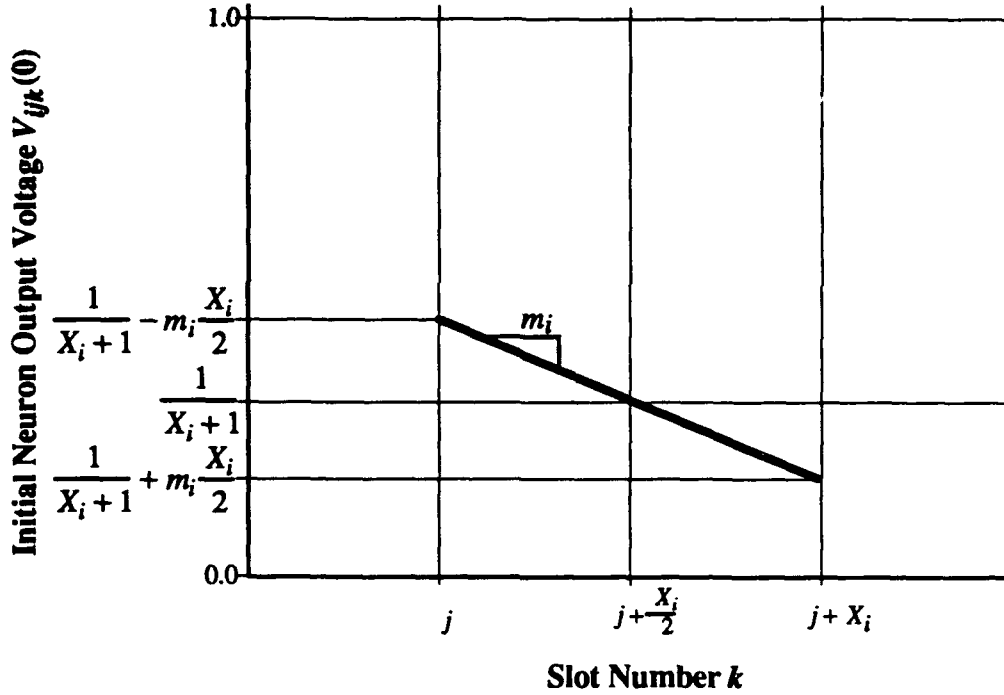


Figure 7.2. Initial neuron output voltage as a result of skewed initialization

To keep the initial output voltages within the output voltage limits, the slope is bounded by $|m_i| \leq 2/(X_i(X_i+1))$. Note that, under this function if the slope m_i is within its bound for all i ,

$$\sum_{k=j}^{j+X_i} V_{ijk}(0) = 1.0$$

as desired. Thus, if $-2/(X_i(X_i+1)) < m_i < 0$, the initial instantaneous state will declare all neurons ijj , $i = 1, \dots, N_{sd}$; $j = 1, \dots, L(i)$, to be "on" because these neurons will have the largest output voltages. In other words, for $m_i < 0$, the j^{th} link in the path is activated in the j^{th} slot of the cycle, which is the first slot in which it could possibly be activated without violating the sequential-activation requirement. Alternatively, for a positive slope ($m_i > 0$), all neurons $ij(j + X_i)$ will initially be "on;" i.e., each link is activated in the last possible slot. Either choice, if used consistently for all the links on the same path, will provide an initial state (based on the instantaneous state description) that is free of sequence conflicts. However, this initial state is not necessarily free from primary conflicts. Typically, after this type of initialization, the NN must remove a large number of primary conflicts. As the number of primary conflicts are reduced, temporary sequence conflicts are often generated. However, simulation results have shown that the discovery of a conflict-free schedule is much more likely when the initial instantaneous state is free of sequence conflicts.

It might be beneficial to use a positive value of m_i for some paths, and a negative value for others. This approach would reduce the expected number of primary conflicts that occur in the initial instantaneous state without creating any sequence conflicts. In this approach, in the initial instantaneous state, the neurons representing the last slot in which links in the paths with $m_i > 0$ can be legally activated are declared to be "on," and the neurons representing the first slot in which links in the paths with $m_i < 0$ can be legally activated are also declared to be "on." Thus, the "tentative" activations of the initial instantaneous state are split between the early and the late slots. A better approach, which is fully discussed in Section 7.3, would be to randomly distribute the tentative activations over the entire set of legal slots for each link, rather than clustering them in the early and/or late slots.

We have developed three different approaches for assigning a value to m_i . In the approach that we refer to as the "conditionally-fixed-value" approach, the value of m_i is given by

$$m_i = \begin{cases} m_c, & |m_c| \leq \frac{2}{X_i(X_i + 1)} \\ \frac{\pm 2}{X_i(X_i + 1)}, & \text{otherwise} \end{cases} \quad (11)$$

Under this method, m_i is set to a fixed constant value m_c for paths that are sufficiently short (i.e., such that $X_i(X_i + 1) \leq 2/|m_c|$), whereas it varies for longer paths. In particular, with $\Lambda = 9$ and $m_c = -0.1$, this approach gives an initial instantaneous state in which the first legal slots of longer paths (paths longer than 5 hops) are activated and the last legal slots of shorter paths are activated. This occurs because there are fewer slots in which a link in a longer path may be legally activated. Therefore, $m_i = m_c$ is used in Eq. (10) as it assigns positive initial values to all neurons representing each link on the longer paths, as shown in Fig. 7.2. In the shorter paths (≤ 5 hops), there are more slots in which each link may be legally activated, and the slope bound must be applied. Thus, for the SD pairs connected by these short paths, Eq. (10) assigns a value of $V_{ij(j+X_i)} = 0.5$ to the neurons that represent the last slots in which links of the path connecting SD pair i may be legally activated, because $m_i = -2/(X_i(X_i + 1))$. This approach tends toward initial overactivation because all of the links ij that form a "short" path (in this example, any path i with $L(i) \leq 5$ hops) have a corresponding set of neurons ijk that have

$$\sum_{k=j}^{j+X_i} V_{ijk}(0) = 1.5.$$

Such overactivation can result in severe oscillatory behavior, which is discussed in Section 7.5.2.

In the second approach, which we call the “path-length-dependent” approach, the value of m_i depends on the path length as follows:

$$m_i = \frac{\theta}{X_i(X_i + 1)},$$

where $|\theta| \leq 2$ is a parameter that controls the amount of skew; a value of $\theta = \pm 2$ gives maximum skew, and a value of $\theta = 0$ provides no skew. We have used $\theta = -1$ so that the slope of the neuron output voltage initialization function is one half of the slope limit for every SD pair, regardless of its path length. Thus, the case of ≤ 0 in Eq. (10) is avoided, eliminating the need to assign initial output voltages values of 0.5.

A third approach, which we call the “hybrid approach” uses concepts from both the conditionally-fixed-value approach and the path-length-dependent approach. In this approach, the value of m_i is given by the following simple modification of the conditionally-fixed-value formulation of Eq. (11):

$$m_i = \begin{cases} m_c, & |m_c| < \frac{2}{X_i(X_i + 1)} \\ \frac{-m_c \theta / |m_c|}{X_i(X_i + 1)}, & |m_c| \geq \frac{2}{X_i(X_i + 1)} \end{cases}.$$

By keeping the slope m_i within its bound, this modification eliminates the overactivation problem associated with the conditionally-fixed-value approach. When the magnitude of the conditionally-fixed slope value m_c exceeds the bound, a path-length-dependent slope value is used with the opposite sign to that of m_c . This prevents the assignment of an initial neuron output-voltage value of 0.5 (i.e., the ≤ 0 case in Eq. (10)), while favoring the activation of links in “long” paths in their first legal slots, and favoring the activation of links in “short” paths in their last legal slots as was done in the original conditionally-fixed-value approach.

7.3 Skewed Randomization

Skewed initialization, which was discussed in Section 7.2, provides an initial instantaneous state that is free of sequence conflicts, but probably contains primary conflicts. Because skewed initialization causes the tentative activation (i.e., the initial activations declared by the instantaneous state) of all of the neurons that represent the first (or the last) slot in which a link may be legally activated, the expected number of primary conflicts in the initial instantaneous state is relatively high. As suggested in Section 7.2, this number can be reduced by randomly distributing the

tentative activations (i.e., the initial activations declared by the instantaneous state) over the entire set of neurons that represent each link. In this section, we present a method, called "skewed randomization," that uses a different form of NN perturbation than the one described in Section 5.0 in conjunction with skewed initialization, to provide an initial instantaneous state that is free of sequence conflicts, and to reduce the expected number of primary conflicts in the initial instantaneous state.

With the skewed randomization method, we randomly select one neuron to be tentatively activated (i.e., to be assigned the largest initial output voltage of any of the neurons ijk , $k = j, \dots, j + X_i$; and hence to be declared "on" by the instantaneous state interpretation) out of the $X_i + 1$ neurons that represent link ij , subject to the constraint that no sequence conflicts are created. The set of neurons that represent link $i1$ (i.e., the link incident on the source node of SD pair i) is comprised of neurons $\{i1k : 1 \leq k \leq 1 + X_i\}$. Of these neurons we randomly select neuron $i1k'$ to be tentatively activated so that the instantaneous state will declare link $i1$ to be active in slot k' . For all links that are not incident on the source node, the set of neurons from which neuron ijk' — the neuron to be tentatively activated for link ij — is selected must be chosen so as to prevent the introduction of sequence conflicts into the initial instantaneous state. Therefore, if $j > 1$, we randomly select neuron ijk' from the set $\{ijk : \max\{j - 1, k' \text{ selected for link } i(j - 1)\} < k \leq j + X_i\}$. Then we apply the modified path-length-dependent initialization given by

$$V_{ijk}(0) = \frac{1}{X_i + 1} + m_i \left((k + k' - 2j) \bmod (X_i + 1) - \frac{X_i}{2} \right). \quad (12)$$

Comparing the initial output voltages given by this method of skewed randomization to those obtained using the path-length-dependent form of skewed initialization discussed in Section 7.2, with $m_i < 0$ in both cases, we find that

$$\begin{array}{ccc} V_{ij(j+n)}(0) & = & \begin{cases} V_{ij(k'+n)}(0), & k' + n \leq j + X_i \\ V_{ij\{(k'+n) \bmod (X_i) + j\}}(0), & k' + n > j + X_i \end{cases} \\ \text{skewed initialization} & & \text{skewed randomization} \end{array}$$

for $n = 0, \dots, X_i$. Thus, the same set of output voltage values are obtained in both cases; they are simply applied to different neurons. With skewed randomization, the initialization provided by randomly selecting the neurons ijk' replaces the additive random perturbation that was used in all of our other NN models.

7.4 Evaluation of Skewed Initialization and/or Randomization via Monte-Carlo Simulation

The effects of skewed initialization and/or skewed randomization on the performance of the reduced SAS NN model were evaluated by means of five Monte-Carlo simulations. Table 7.4 summarizes the results of, and the differences between, these five runs. The parameter values that were common to all of the runs are shown in Table 7.5. Each of the runs attempted to find 9-slot sequential-activation schedules (optimum schedules) that satisfy the requirements of Table 2.1 and Fig. 2.2 from 100 different initial states.

In Table 7.4, an entry of "perturbation" in the randomization column means that the run used the standard method of randomizing the initial state (which involves adding a random quantity to each of the neuron's initial input voltages, as described in Section 5.0). The initialization and the " m_c or θ " columns describe the type of initialization and the initialization parameter values that were used.

Table 7.4. Differences between the simulations using skewed initialization and/or randomization

Run	randomization	β	initialization	m_c or θ	T_o	τ_T	G_{end}	% optimal
D	perturbation	1	conditionally fixed-value	$m_c = -0.1$	off	off	off	79
D'	perturbation	1	hybrid	$m_c = -0.1$ $\theta = 1.9$	off	off	off	64
E	perturbation	1	path-length dependent	$\theta = -1$	0.01	50	10^4	7
F	skewed	1	path-length dependent	$\theta = -1$	0.01	50	10^4	9
G	skewed	β_n	path-length dependent	$\theta = -1$	0.01	50	10^4	21

Table 7.5. Parameters values used in all simulations using skewed initialization and/or randomization

$\lambda_c(0)$	$(\Delta t)_{\lambda_1}$	$(\Delta t)_{\lambda_{2ij}}$ $(\Delta t)_{\lambda_3}$	$(\Delta t)_{\lambda_4}$	Δt	I	N_{i-max}	N_c	ζ	Λ	MLM
1	0.02	0.1	0.01	10^{-4}	10	11000	10^4	0.75	9	yes

In Run D, the conditionally-fixed-value approach was used in the absence of simulated annealing. All of the remaining parameter values were the same as those used in Run C (see Section 7.1), which found optimal schedules in 7% of its simulations. As noted in Table 7.4, Run D found optimal schedules in 79% of the runs. Thus, the percentage of optimum schedules found by the NN is an order of magnitude larger when this form of skewed initialization is used. Within

the set of 79 optimal solutions found in Run D, there are 39 different schedules. The fact that many different optimal schedules were found confirms that the use of different random seeds results in the search of different regions of the state space. The average number of iterations required to find an optimal schedule, given that an optimal schedule was found, was 2888 iterations; the average number of iterations to termination, including the runs that failed to remove all conflicts, was 5598 iterations.

As can be seen in Table 7.4, Run D gave the best performance by far. However, the use of the conditionally-fixed-value form of skewed initialization may cause initial neuron overactivation, as noted in Section 7.2. Such overactivation can result in severe oscillatory behavior, which is discussed in Section 7.5.2. This problem can be eliminated by using the hybrid form of initialization. In a Monte-Carlo simulation using hybrid initialization with $\theta = 1.5$ and the parameter values of Run D, 39% of the schedules found in simulations from 100 different states were optimal. Increasing the value of θ to 1.9 in Run D' produced a total of 64 optimal schedules in 100 attempts. The percentage of optimal schedules produced using the hybrid form of initialization is not as high as that produced using the conditionally-fixed-value form, but it does yield nearly the same performance without the danger of overactivation. However, multiple-instance simulations, which will be discussed further in Section 7.5, have indicated that neither the hybrid, nor the conditionally-fixed-value form of initialization provides the consistently good performance that is obtained when using the path-length-dependent form of initialization is used to schedule a number of different problem instances.

Runs E, F, and G were performed in an effort to discover a NN model capable of delivering good performance while using the path-length-dependent form of skewed initialization. This form of initialization avoids the overactivation problem while, hopefully, yielding the desired effects. In Run E, the use of the path-length-dependent approach in conjunction with GSA yielded only 7% admissible solutions. This is the same fraction as was obtained in Run C where, in essence, m_i was set to 0 for all i . Both Runs F and G used the path-length-dependent approach to initialization in conjunction with skewed randomization. In both runs, the NN evolution in the presence of GSA resulted in improved performance over Run E. The parameters for both Runs F and G are identical with the exception of β . Recall that β is a coefficient that adjusts the number of transmissions N_x (neuron activations) required by the third constraint. In Runs D, E, and F, the parameter β was set to one. Run G, however, used the nonmonotonically increasing function β_n , which was discussed in Section 6.4. When the use of $\beta < 1$ was introduced in the NAS simulations, significantly improved performance was noted. If Run D is disregarded, the same holds here; the percentage of optimum schedules found by the NN using β_n was more than twice

as large as the percentage found when using a constant, unit-valued β . Although it does not match the performance of Run D, Run G demonstrates that path-length-dependent initialization may be used to obtain reasonably good results without danger of initially overactivating the NN and thus causing oscillatory behavior.

7.5 Evaluation of Skewed Initialization and/or Randomization via Multiple-Instance Simulation

Multiple-instance simulations of the reduced SAS NN model were performed to evaluate the use of skewed initialization and/or randomization. These simulations scheduled the 50 problem instances in set (7, >7) that are listed in Table A5 in Appendix A. Recall that each of the problem instances in set (7, >7) has $B_{nas} = 7$ as a lower bound on its nonsequential-activation schedule length and a NAS heuristic schedule length greater than 7 slots. Although these problems instances were selected on the basis of a nonsequential-activation scheduling analysis of the network of Fig. 2.1 and of the paths listed in Table A1 in Appendix A, they represent a set of problem instances that offers a degree of diversity while maintaining some common characteristics. Of the 377 path sets that are elements of the set (7, >7), 233 have lower bounds on the sequential schedule length of $B_{sas} = 8$ slots, 114 have $B_{sas} = 9$ slots, and the remaining path sets have $B_{sas} = 10$ slots. The SAS heuristic found schedules for these specifications with lengths ranging from 9 slots to 14 slots. The majority of the schedules generated by the heuristic were 10 or 11 slots long; 25 were 9-slot schedules, and 5 were 14-slot schedules. None of the heuristic schedules were as short as their corresponding bound.

Before presenting the results of these simulations in Sections 7.5.2 and 7.5.3, concerns that are unique to SAS multiple-instance simulation are discussed in Section 7.5.1. In Section 7.5.4, we discuss the results of multiple-instance simulations that scheduled a set of more heavily-constrained problem instances.

7.5.1 SAS Multiple-Instance Simulation Issues

Since Λ^* (the shortest possible schedule length for a particular problem instance) is not known a priori, an attempt is first made to generate a schedule of length $\Lambda_0 = B_{sas}$. If an admissible schedule has not been found after N_{s-max} attempts, the value of Λ is incremented by one and up to N_{s-max} attempts are made again. This process is repeated until an admissible solution is found. The parameter N_{s-max} is critical to both the quality of the solution and to computational efficiency. Clearly, none of the runs for which $\Lambda < \Lambda^*$ can possibly produce an admissible schedule. Thus, use of an excessively large value of N_{s-max} results in a large number of futile

runs. On the other hand, use of too small a value of N_{s-max} can result in the failure to find a schedule of optimal length. For example, increasing the value of N_{s-max} from 5 to 20 in a pair of otherwise identical multiple-instance SAS simulations permitted the generation of shorter schedules for a significant number of problem instances; the percentage of schedules that were no longer than those generated by the SAS heuristic was increased from 57.4% to 90%. Thus, although the NN is able to find an admissible schedule fairly rapidly if $N_{s-max} = 5$ (because the schedule length is increased after only five unsuccessful attempts), it is also likely to overlook admissible schedules with shorter length. Increasing the value of N_{s-max} to 20 significantly increases the probability of discovering a "short" schedule.

We have not been able to determine the length of an optimum sequential-activation schedule for many of the problem instances in the set (7, >7). When a minimum schedule length has been ascertained for a problem instance, confirmation has been achieved almost exclusively by means of the discovery of a NN schedule with length that matches the tightened bound (i.e., for a few problem instances, we have been able to increment B_{sas} by examination of the network; e.g., see Appendix B). Therefore, in the following subsections, when we speak of the percentage of minimum-length schedules, we are actually referring to the subset of problem instances that have been scheduled by the NN and for which the minimum schedule length is known. To assess the quality of all of the admissible schedules generated in a simulation, including those with unknown minimum length, we compare their lengths to the lengths of the schedules found by the SAS heuristic.

7.5.2 *Evaluation of the Conditionally-Fixed-Value and the Hybrid Form of Initialization*

A multiple-instance simulation, labeled Run H, was programmed to schedule the 50 problem instances in set (7, >7) that are listed in Table A5 in Appendix A. Run H uses nearly the same NN model as was used in Run D, i.e., the model that yielded the best performance in the Monte-Carlo simulations (see Section 7.4). In Run D, the conditionally-fixed-value form of skewed initialization was used, and the NN was iterated in the absence of simulated annealing. Run H differs in that it does apply Gaussian simulated annealing, and that it does not deliver the same quality of performance as was obtained in Run D. However, the performance degradation is not caused by the use of GSA; it is actually a result of the conditionally fixed-value form of skewed initialization that was used in both runs.

In Run H, the parameter N_{s-max} , that is the number of different initial states that must be simulated before incrementing Λ , was set equal to five. Admissible nonoptimal schedules were successfully found for the first nine problem instances listed in Table A5. Five of these schedules

were shorter, and two were longer, than the schedule generated by the SAS heuristic for the corresponding problem instance. In attempting to schedule the tenth problem instance, the NN state became oscillatory, and the run was terminated.

Explanation of Oscillatory Behavior

In the Monte-Carlo simulations of Run D, the value of Λ was fixed at 9 slots. In the multiple-instance simulations of Run H, Λ_0 is set to B_{sas} and Λ is incremented until an admissible schedule is found. As the value of Λ is increased as a result of the failure to find an admissible schedule within N_{s-max} ($= 5$) attempts from different initial states, the number of SD pairs for which $|m_c| = 0.1 > 2/(X_i(X_i + 1))$ also increases. In turn, a greater number of neurons are assigned an initial output voltage of 0.5. In some cases this presents no particular problems, and, as indicated by the results of Run D and the first nine solutions of Run H, actually works fairly well until the number of neurons that are assigned initial output voltages values of 0.5 exceeds some problem-dependent threshold. It appears that when the threshold is exceeded, the high average initial activation level of the NN causes excessive growth of the LM and, in short order, results in an oscillatory state where the output voltages of all of the neurons is alternately 0 and 1 at each iteration.

This behavior was exhibited when Run H attempted to schedule the tenth problem instance listed in Table A5.¹⁴ After failing to find admissible 8-, 9-, or 10-slot schedules, but exhibiting no oscillatory tendencies, the oscillation threshold was exceeded when the 11-slot NN was initialized. Simulations from five different initial states with 11, 12, and 13 slots all became oscillatory within 15 iterations. The tenth problem instance requires a total of 45 transmissions (i.e., $N_x = 45$). Two of the paths specified by this problem instance are 7 hops long, the remaining eight paths are shorter. With $m_c = -0.1$ and $\Lambda = 10$, the neurons that represent the last slot in which links in paths shorter than 7 hops may be legally activated will be assigned an initial output voltage of 0.5. Thus, the sum of the output voltages of the neurons that represent each of the 31 links in paths shorter than 7 hops is equal to 1.5 (neglecting the perturbations). When $\Lambda = 11$, the neurons that represent the last slot in which the links in paths shorter than 8 hops may be legally activated are assigned an initial output voltage of 0.5. Since all of the paths in this problem instance are shorter than 8 hops, all of the links are initially favored in their last legal slot, and the corresponding neurons are assigned an initial output voltage of 0.5. Thus, the sum of the output voltages of the

¹⁴ The length of the optimum schedule for this problem instance is not known. We have been able to increment B_{sas} (see Appendix B) so that the greatest known lower bound is 9 slots. The shortest admissible schedule that has been found is 10 slots.

sets of neurons that represent each of the links is equal to 1.5, resulting in an excessive initial NN activation level.

In an effort to duplicate the performance of Run D, a rerun of Run H without GSA was also tried. The results were the same as those from Run H through the third problem instance. However, in the simulation that attempted to schedule fourth problem instance listed in Table A5, the oscillatory behavior — that was not apparent until the tenth problem instance in Run H where GSA was used — was once again observed. The conclusion is that, in this case, the use of additive noise may be helpful in delaying the onslaught of oscillations by coincidentally lowering the average activation level in the NN.

A Simulation Using Hybrid Initialization

A third multiple-instance simulation, labeled Run I, was performed using the hybrid form of initialization (with $m_c = -0.1$, $\theta = 1.9$, and $N_{s-max} = 20$; the remaining parameter values were the same as those used in Run H) in the absence of simulated annealing. The fact that this run was successful in finding schedules for all of the problem instances listed in Table A5, including those which caused Run H to become oscillatory, verifies that hybrid initialization eliminates the problem of oscillations that result from initial overactivation. Although admissible schedules were found consistently by Run I, only 30% of the schedules were shorter than those found by the SAS heuristic, and 36% were longer than the schedules generated by the heuristic. Seven of the 50 schedules produced in Run I are known to be optimal (i.e., minimum in length), and 35 are known to be nonoptimal.¹⁵ Thus, the good performance that was obtained in the Monte-Carlo Run D' (Section 7.4), and the relatively poor performance obtained in this multiple-instance simulation, indicate that the performance of hybrid initialization is inconsistent.

The results of Runs D and H indicate that the conditionally-fixed-value form of initialization may work extremely well for specific problem instances, but its tendency towards overactivation may result in severe oscillation and in the inability to find an admissible schedule for other problem instances. Run I demonstrates that the overactivation problem may be avoided by using hybrid initialization, but the resulting schedules are not uniformly good. In the next subsection, simulations show that, in general, significantly better results are obtained using the path-length-dependent form of initialization.

¹⁵ The schedules that have been verified to be optimal have lengths equal to a tightened lower bound on the schedule length (see Appendix B). The schedules that have been verified to be nonoptimal are longer than an admissible schedule generated in a different simulation.

7.5.3 Evaluation of the Path-Length-Dependent Form of Initialization

We have evaluated the use of the path-length-dependent form of initialization in a number of variants of the reduced SAS model by means of multiple-instance simulations that scheduled the problem instances listed in Table A5. The best results were found using the same NN model that was used in Run G of Section 7.4. In addition to using path-length-dependent initialization, this model employed GSA, skewed randomization, and the nonmonotonic function β_n . In this simulation, the value of N_{s-max} was set equal to 20; the remaining parameter values are listed in Tables 7.4 and 7.5. Only two of the 50 schedules found by this model were longer than the schedules found by the SAS heuristic. Eleven of the schedules have been shown to be optimal; thirteen of the schedules are known to be nonoptimal. Six of the certified nonoptimal schedules have length $\Lambda^* + 1$ (for these problem instances, we have been able to ascertain that $\Lambda^* = 9$). Another five were verified to be nonoptimal by shorter admissible schedules that were generated by a second simulation of the same NN model with $N_{s-max} = 40$ (instead of 20). We know that the remaining two nonoptimal schedules have lengths greater than Λ^* because the schedules generated by the SAS heuristic are shorter than those found by the NN. The fact that, for certain problem instances, the NN found schedules whose lengths matched a lower bound¹⁶ on the sequential schedule length was used to ascertain the minimum schedule length Λ^* for those communication requirements.

This simulation serves to confirm the conclusions drawn in Section 7.4 and Section 7.5.2. The NN model yielded better performance than the SAS heuristic (whereas 22% of the NN schedules have been verified to be optimum and 74% *may* be optimum, only 26% of the heuristic schedules *may* be optimum but none have been verified; 64% of the NN schedules were shorter than the corresponding schedules generated by the SAS heuristic) without the risk of incurring an oscillatory NN state. As found in nonsequential-activation scheduling (Section 6.5), and in Section 7.4, the use of the nonmonotonic function β_n also is beneficial to the NN performance. Furthermore, this simulation demonstrates the model's ability to reliably generate sequential-activation schedules of minimum, or nearly minimum, length for a diverse set of problem instances.

¹⁶ Typically, this bound was $B_{sas} + 1$ slots, because, as discussed in Appendix B, it was found by contradiction that the network could not be scheduled in B_{sas} slots. Therefore the bound was tightened to $B_{sas} + 1$ slots.

7.5.4 SAS of the Problem Instances with Known Minimum Schedule Lengths of 8 Slots

As discussed in Section 3.3, the SAS heuristic was able to find optimal 8-slot sequential-activation schedules for the eight path sets listed in Table A4 in Appendix A. These problems are more-highly constrained than the problems for which 9-slot schedules are needed, and present a significant challenge to our methodology. We now discuss the use of our SAS NN model to schedule the sequential activation of the links in these path sets. In a multiple-instance simulation of this set of problem instances, it was reasonable to set N_{s-max} to a large value ($N_{s-max} = 500$) because the minimum schedule length is known for these problem instances, i.e., because $B_{sas} = \Lambda^*$. The NN model that was used in this multiple-instance simulation is the same as the one used in Section 7.5.3; it used skewed randomization, the nonmonotonic function β_n , and GSA.

The large value of N_{s-max} allowed the NN to find optimal schedules for each of the problem instances. However, an average of 85.5 different initial states were required to find each of these schedules. Four of the optimal schedules were each found within the first 25 different initial states, 40 different starting points were required to optimally schedule one of the problem instances, and two of the problem instances each required 100 different initial states. An optimal schedule was not found for the eighth problem instance until 352 different initial states had been tried. This schedule is shown in Table 7.6.

In the schedule shown in Table 7.6, there are only 7 “open” cells out of the 192 cells containing an entry (each of which represents a neuron). Thus, the schedule shown in Table 7.6 is a 96.4% maximal schedule. The fact that this schedule is nearly maximal indicates that this particular problem instance is highly constrained. (Typical 9-slot scheduling problems, even in cases for which $\Lambda^* = 9$, are not as highly constrained.) Thus we have again demonstrated the capability of our NN problem formulation to solve highly-constrained problems.

The results of this simulation again demonstrate the ability of the SAS NN model to produce optimal sequential-activation schedules. Since the SAS heuristic was also able to produce optimal schedules for these eight path sets, we were not looking for improved performance, but rather to see whether the SAS NN model was able to perform as well as the SAS heuristic. Actually, the SAS heuristic was more efficient in the sense that many runs from different initial states were often needed for the SAS NN model to find an optimal schedule. However, because the communication requirements of Table A4 require nearly maximal schedules, we view this more as an indication of good performance by the SAS heuristic for these particular examples, than of poor SAS NN performance. Of course, the true power of the SAS NN model is more apparent in those examples that it can schedule optimally, but which the SAS heuristic cannot.

Table 7.6. An optimal sequential-activation schedule for the eighth path set of Table A4
(X = Link activation, o = open for activation, b = link blockage,
i = ineligible for activation owing to the blockage of an adjacent link)

Path [†]	Link	Indices (SD, link)	Slot							
			1	2	3	4	5	6	7	8
0	(4->5)	1,1	b	b	b	X	b			
	(5->13)	1,2		b	b	b	X	b		
	(13->20)	1,3			b	b	b	X	b	
	(20->24)	1,4				i	i	b	o	X
6	(7->14)	2,1	X	b	b	b	b	b		
	(14->15)	2,2		X	b	b	b	b	b	
	(15->17)	2,3			X	b	b	o	b	b
11	(9->12)	3,1	X	b	b	i				
	(12->7)	3,2		b	X	b	b			
	(7->14)	3,3			b	b	X	b		
	(14->15)	3,4				b	b	b	X	
	(15->16)	3,5					b	i	b	X
12	(1->4)	4,1	b	o	X	b	o			
	(4->5)	4,2		b	b	b	b	X		
	(5->13)	4,3			b	b	b	b	X	
	(13->19)	4,4				b	b	b	b	X
22	(5->13)	5,1	X	b	b	b	b	b		
	(13->12)	5,2		X	b	b	b	b	b	
	(12->11)	5,3			b	o	X	o	b	b
32	(21->22)	6,1	X	b	i	i				
	(22->20)	6,2		X	b	i	i			
	(20->13)	6,3			X	b	b	b		
	(13->7)	6,4				X	b	b	b	
	(7->6)	6,5					b	b	b	X
36	(1->2)	7,1	X	b	b					
	(2->3)	7,2		X	b	b				
	(3->6)	7,3			X	b	b			
	(6->8)	7,4				X	b	b		
	(8->9)	7,5					X	b	b	
	(9->10)	7,6						X	b	b
43	(3->6)	8,1	X	b	b	b				
	(6->5)	8,2		X	b	b	b			
	(5->14)	8,3			X	b	b	b		
	(14->15)	8,4				X	b	b	b	
	(15->18)	8,5					X	o	b	b
48	(2->3)	9,1	b	b	b	X				
	(3->6)	9,2		b	b	b	X			
	(6->8)	9,3			b	b	b	X		
	(8->9)	9,4				b	b	b	X	
	(9->12)	9,5					b	b	b	X
50	(14->7)	10,1	b	b	b	b	b	X		
	(7->11)	10,2		i	b	b	b	b	X	
	(11->8)	10,3			i	b	b	b	b	X

[†] See Table A1 in Appendix A.

7.6 Conclusions on the SAS NN Model

The results of this section have demonstrated the ability of the NN model to find minimum- or nearly minimum-length sequential-activation schedules for several difficult problem instances. They also indicate that the NN model is robust and may perform well in a broad class of scheduling problems.

For example, the minimum length of a sequential-activation schedule that satisfies the communication requirements of Table 2.1 is 9 slots. All of the SAS NN models have been able to find such a schedule, with varying degrees of efficiency. In Monte-Carlo simulations that scheduled the sequential activation of the links in this problem instance, the best performance was obtained in Run D (see Section 7.4), which produced 79% optimal solutions, and which used the same NN model as was used in the multiple-instance Run H (Section 7.5.2). However, the model used in Section 7.5.3, which is the same as that used in Run G of Section 7.4, gave the best performance in the multiple-instance simulations. From these observations, we conclude that the NN model that was used in Run G gives the best overall performance when examining diverse problems. However, one of the other models might be better suited for a given problem instance; e.g., the NN model that was used in Run E (Section 7.4) gave fairly poor results when scheduling the requirements of Table 2.1 in Monte-Carlo simulations, but, in contrast, was the only model that found optimal (9-slot) schedules for the 4th and 13th problem instances listed in Table A5 in Appendix A.

8.0 THE JOINT ROUTING-SCHEDULING PROBLEM

In all types of communication networks, it has been a common practice to break up the enormous total network design problem into subproblems, each of which can be studied in isolation. This partitioning usually follows the “layered” structure of the OSI architecture (see e.g., [29]). Thus, even though it is recognized that problems, issues, and design choices that reside in separate layers are, in fact, interdependent, they are addressed separately; only at the final “integration” stage is there an occasional attempt to recognize their influence on each other.

However, it is increasingly being recognized that in certain cases the interaction between two or more factors from different layers may be so fundamental and strong that their joint effects must be studied simultaneously to provide nearly-optimal, or even merely acceptable, performance. In particular, in multihop radio networks the primary control issues are channel access and routing, which are, in fact, intimately related to each other.

In this report we have addressed the use of contention-free link activation as the mechanism for channel access. The link-scheduling function determines the capacities of the links in the network,¹⁷ one of the most critical factors in the generation of routing schemes. Conversely, the routing scheme impacts directly on the traffic that must be supported over each of the network's links.

Despite the intimate relationships that exist between these network control mechanisms, few attempts have been made to address them jointly, e.g., see [6] and [9]. The formulation in these papers is based on a continuous traffic model, in which packets are in effect subdivided into infinitesimal pieces, and therefore does not apply directly to the case of discrete packets. Additional background information on the joint routing-scheduling problem is provided in [15].

As a partial solution, a NN approach developed in [15, 16] starts with a set of multihop paths between each of several SD pair. It attempts to pick those paths that result in "small" numbers of shared links amongst the chosen paths so that "congestion" at these links is reduced, permitting the generation of relatively short schedules. In such a formulation there is an objective function (reflecting the desire to minimize congestion), but the slots in which the links are to be activated are not determined. This method was, in fact, used to select the unique paths such as those enumerated in Table 2.1, which are used to determine the communication requirements for the problem instances considered in this report. Minimization of congestion was chosen as the performance measure because it was hypothesized that doing so would permit the generation of short schedules. As we now discuss, this was not strictly true, but the schedule length required for path sets with low congestion tended to be lower than that for path sets with high congestion.

To evaluate the hypothesis that minimization of congestion would permit the generation of short schedules, the "congestion energy" of each of the 858 path sets that admit 7-slot nonsequential-activation schedules (the problem instances in the union of sets (7, 7) and (7, >7)) was calculated. Congestion energy is the measure of the total number of shared links in the network, taken on a pairwise basis, that was used in [15, 16] as a metric for the solutions to the routing-to-minimize-congestion problem.¹⁸ In that study, a routing NN model, in which one neuron was defined to correspond to each path, found solutions to the routing problem given by Table A1 in Appendix A with congestion energy values of 33.75 in each of 100 runs from different initial states. This was actually the second lowest congestion energy possible; the minimum congestion-energy value was 33.25. A second, more complex routing NN model, in which one

¹⁷ The number of times a link is activated per frame determines the capacity of that link.

¹⁸ This is the modified congestion energy that is discussed in Section 4.9 of [15].

neuron was defined to correspond to each link, was also considered. Although this "link-neuron" model was unable to find low congestion-energy solutions as frequently as the "path-neuron" model, it did find sets of paths with the minimum congestion-energy value of 33.25 in 3 of 100 runs from different initial states. A different version of the link-neuron model yielded more consistent performance, although it was unable to discover any minimum congestion-energy solutions. This version found solutions with the second lowest congestion-energy value (33.75) in 17 of 100 runs from different initial states.

In Fig. 8.1, the cumulative mass function of these results is compared to that obtained by considering all permutations of the paths in Table A1 (exhaustive search), and to that obtained by considering all permutations of the subset of paths in Table A1 that consists of only the shortest paths between each SD pair (shortest-path heuristic). The figure shows that the path sets that admit minimum-length nonsequential-activation schedules do, indeed, tend to have low congestion energy. However, the relationship is not monotonic: The 12 path sets with minimum congestion energy (33.25) do, in fact, admit 7-slot schedules, but only 12 of the 30 path sets that have congestion energy values of 33.75 admit 7-slot schedules. The congestion energy of the 858 path sets that can be scheduled in 7 slots ranges from 33.25 to 46.25. However, there are a total of 498,975 path sets that have congestion energy in this range; thus, there are many path sets that cannot be scheduled in 7 slots, but do have lower congestion-energy values than some of the path sets that can be scheduled in 7 slots. The existence of relatively high congestion-energy solutions that *do* admit optimum schedules, and of relatively low congestion-energy solutions that *do not* admit optimum schedules indicates that the routing and the scheduling problems are not separable.

From a different viewpoint, we have found that the shortest-path heuristic generally yields a set of routes with reasonably low congestion energy. For the problem posed by Table A1, the set of paths chosen by the shortest-path heuristic (i.e., the path set with the lowest congestion energy of any of the path sets in the restricted subset that includes only the shortest paths between each SD pair) has a lower congestion-energy value than 56% of the path sets that can be scheduled in 7 slots. However, it turns out that *none* of the path sets considered by the shortest-path heuristic can be scheduled in 7 slots. In fact, the minimum-length nonsequential-activation schedule for any of these path sets is 9 slots.

Although a path set that has a low congestion-energy value will probably admit a short nonsequential-activation schedule, the above observations lead to the conclusion that the use of congestion minimization as the sole criterion for route selection is inadequate to guarantee a solution that can be optimally scheduled. However, for the routing-scheduling problem given by Table A1, minimizing the maximum nodal degree in the network does yield a set of paths that

admits an optimal (7-slot) schedule. This phenomenon results from the absence — in any of the 858 path sets that have maximum nodal degrees of seven — of odd-length cycles that require more than seven slots to schedule. As discussed in Section 3.1.1, odd-length cycles may require more slots in the schedule than do the maximum-degree nodes. Since verifying the presence or absence of odd-length cycles is, in general, excessively computation intensive, simply minimizing the maximum nodal degree is also inadequate to guarantee a solution that can be optimally scheduled because the absence of odd-length cycles cannot be readily verified.

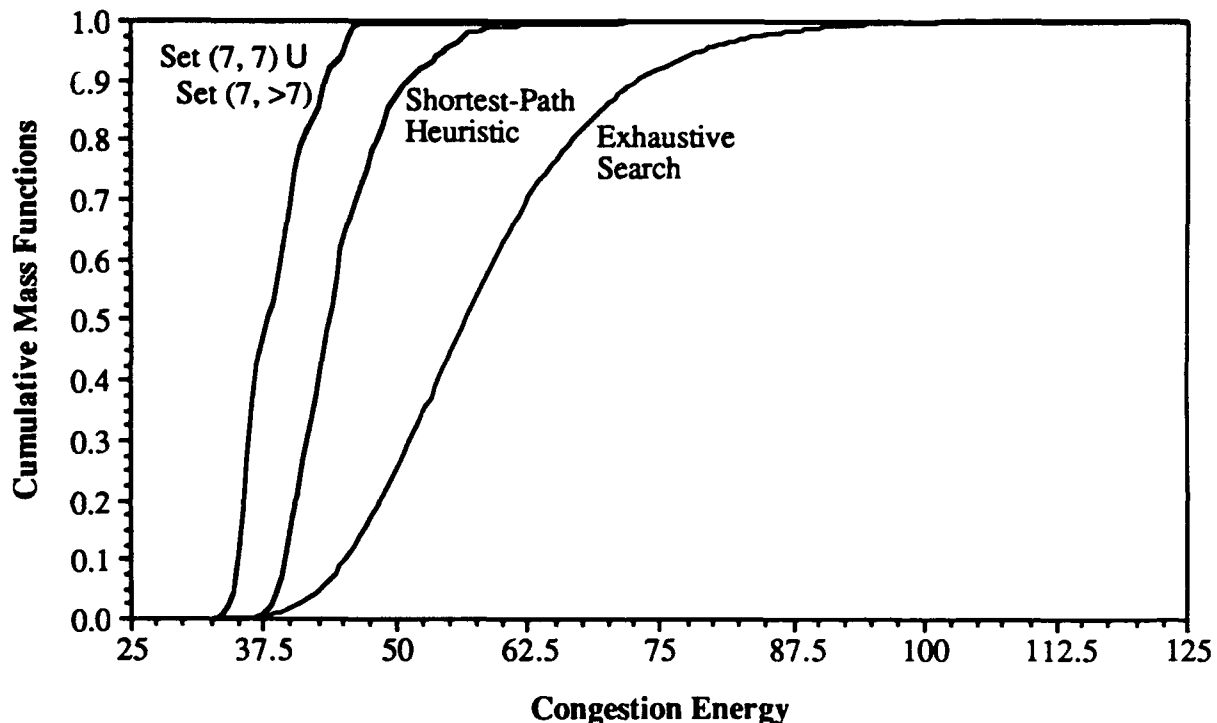


Figure 8.1. Congestion energy of the path sets that admit minimum length schedules.

Thus, selecting routes to minimize congestion energy, or selecting routes to minimize the maximum nodal degree in the network, are both reasonable heuristic methods that tend to yield solutions that admit short, if not optimum, schedules. However, neither approach is capable of delivering solutions that can guarantee minimal values of schedule length. In fact, it appears that any approach that separates the routing problem from the scheduling problem cannot do so. Hence, a joint formulation is needed to simultaneously select routes and schedule link activations optimally.

The SAS problem is a restricted case of the NAS problem that would also benefit from a joint routing-scheduling approach. Because link activations are restricted by the position of the links in the paths, as well as by adjacent links, route selection is even more critical for the SAS problem than it is for the NAS problem. Therefore, a joint routing-scheduling formulation is

necessary, and appropriate, for this problem as well. The joint formulation proposed in the next subsection addresses either NAS or SAS.

8.1 Problem Formulation

Our problem formulation combines the path selection problem addressed in [15, 16] with the scheduling problem discussed in this report. Thus the problem becomes the simultaneous choice of one of these paths for each SD pair along with the determination of a sequential-activation schedule for each link along this path, such that a schedule of minimum length with no scheduling conflicts is produced. Since we are considering discrete packets, a combinatorial-optimization formulation is again appropriate.

In the following subsection we propose a NN model to address the joint routing-scheduling problem. The NN designed for this application is extremely complex, and has not yet been implemented completely. We acknowledge that the model description is not complete; modifications and/or the incorporation of heuristics would almost definitely be needed to produce optimal or nearly optimal results. The model has been included to illustrate the major considerations that must be incorporated into a NN model for the joint routing-scheduling problem.

Problem Statement

Given a set of N_{sd} source-destination (SD) pairs and an equal number of sets of paths, the i^{th} of which contains $N_p(i)$ paths connecting SD pair i , and one unit¹⁹ of traffic to be delivered between each SD pair, select one path from each set of $N_p(i)$ paths that satisfies the communication requirements and schedule the activation of the links in these paths in a minimum number of slots. As was done in addressing the scheduling problem, we may either specify sequential-activation scheduling (SAS) or nonsequential-activation scheduling (NAS).

8.2 A Joint Routing-Scheduling NN Model

We would like to satisfy the specified communication requirements in Λ slots.²⁰ For every link in each of the paths connecting the N_{sd} SD pairs, we define Λ neurons, each corresponding to one slot. We use four integers to index the neurons, i.e., neuron $ijkl$ represents slot l of the k^{th} link in the j^{th} path between SD pair i . As in the pure scheduling problems discussed earlier in this report, the Lyapunov energy function consists only of constraint terms; there is no objective

¹⁹ The model may be easily extended to handle nonunit traffic; e.g., see [15, 16], Section 4.8.

²⁰ Again, we may use either a fixed schedule length or the adjustable-length method (described in Section 4.5.3), under which a penalty is incurred for using the higher-numbered slots.

function to be minimized. We have identified eight constraints that represent the desired objective of the NN for the joint routing-scheduling problem. These constraints incorporate features of the routing constraints developed in [15, 16] as well as the scheduling constraints discussed in this report. In the following, we show the energy expression associated with each of the constraints; in addition we show the corresponding equation-of-motion term, which is found by applying Eq. (7) as was done in Section 4.4 for the pure scheduling problem.

RS₁. Activate (select) links from no more than one path per SD pair:

$$E_1 = \frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{\substack{n=1 \\ n \neq j}}^{N_p(i)} \sum_{k=1}^{L(ij)} \sum_{o=1}^{L(in)} \sum_{l=1}^{\Lambda} \sum_{p=1}^{\Lambda} \frac{V_{ijkl} V_{inop}}{L(in)} = 0,$$

which implies that

$$\frac{-\partial E_1}{\partial V_{ijkl}} = - \sum_{\substack{n=1 \\ n \neq j}}^{N_p(i)} \sum_{o=1}^{L(in)} \sum_{p=1}^{\Lambda} \frac{V_{inop}}{L(in)}.$$

Here $N_p(i)$ is the number of paths between SD pair i , and $L(ij)$ is the length of the j^{th} path between SD pair i .

RS₂. Activate a total of exactly N_{sd} paths in the network:

$$E_2 = \frac{1}{2} \left(\sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{k=1}^{L(ij)} \sum_{l=1}^{\Lambda} \frac{V_{ijkl}}{L(ij)} - N_{sd} \right)^2 = 0,$$

which implies that

$$\frac{-\partial E_2}{\partial V_{ijkl}} = \frac{-1}{L(ij)} \left(\sum_{m=1}^{N_{sd}} \sum_{n=1}^{N_p(m)} \sum_{o=1}^{L(mn)} \sum_{p=1}^{\Lambda} \frac{V_{mnop}}{L(mn)} - N_{sd} \right).$$

Note that this formulation also serves as a constraint that requires the links in selected paths to be activated in one and only one slot, and links in unselected paths to be off at all times.

RS₃. Activate exactly one path per SD pair:

$$E_3 = \frac{1}{2} \sum_{i=1}^{N_{sd}} \left(\sum_{j=1}^{N_p(i)} \sum_{k=1}^{L(ij)} \sum_{l=1}^{\Lambda} \frac{V_{ijkl}}{L(ij)} - 1 \right)^2 = 0,$$

which implies that

$$\frac{-\partial E_3}{\partial V_{ijkl}} = \frac{-1}{L(ij)} \left(\sum_{n=1}^{N_p(i)} \sum_{o=1}^{L(in)} \sum_{p=1}^{\Lambda} \frac{V_{inop}}{L(in)} - 1 \right).$$

This may be easily converted to a MLM constraint. This constraint, like RS₂, also requires that each link in selected paths be activated exactly once.

RS₄. Activate complete paths:

$$E_4 = \frac{1}{2} \left(N_{sd} - \frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{k=1}^{L(ij)} \frac{\sum_{l=1}^{\Lambda} V_{ijkl} \left(\sum_{p=1}^{\Lambda} V_{ij(k-1)p} + \sum_{q=1}^{\Lambda} V_{ij(k+1)q} \right)}{L(ij) - 1} \right)^2 = 0,$$

where we define

$$V_{ij(k-1)l} = \begin{cases} V_{ij(k-1)l}, & k \neq 1 \\ 0, & k = 1 \end{cases} \quad V_{ij(k+1)l} = \begin{cases} V_{ij(k+1)l}, & k \neq L(ij) \\ 0, & k = L(ij) \end{cases}$$

Then, the corresponding equation-of-motion term is given by

$$\begin{aligned} \frac{-\partial E_4}{\partial V_{ijkl}} = & \left(N_{sd} - \frac{1}{2} \sum_{m=1}^{N_{sd}} \sum_{n=1}^{N_p(m)} \sum_{o=1}^{L(mn)} \frac{\sum_{p=1}^{\Lambda} V_{mnop} \left(\sum_{q=1}^{\Lambda} V_{mn(o-1)q} + \sum_{r=1}^{\Lambda} V_{mn(o+1)r} \right)}{L(mn) - 1} \right) \\ & \times \left(\frac{\sum_{q=1}^{\Lambda} V_{ij(k-1)q} + \sum_{r=1}^{\Lambda} V_{ij(k+1)r}}{2(L(ij) - 1)} \right) \end{aligned}$$

It appears that the sequentiality constraint could be, at least partially, enforced by a modified formulation of this constraint given by:

$$E_4 = \frac{1}{2} \left(N_{sd} - \frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{k=1}^{L(ij)} \frac{\sum_{l=1}^{\Lambda} V_{ijkl} \left(\sum_{p=1}^{l-1} V_{ij(k-1)p} + \sum_{q=l+1}^{\Lambda} V_{ij(k+1)q} \right)}{L(ij) - 1} \right)^2 = 0.$$

The use of this modified formulation would serve to reenforce constraint RS_8 below, which is direct extension of the sequentiality constraint used in the pure scheduling NN model.

RS_5 . Activate no conflicting links (no primary conflicts):

$$E_5 = \frac{1}{2} \sum_{i=1}^{N_{sd}} \sum_{m=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{n=1}^{N_p(m)} \sum_{k=1}^{L(ij)} \sum_{o=1}^{L(mn)} \sum_{l=1}^{\Lambda} |A_{ijk} \cap A_{mno}| V_{ijkl} V_{mno l} = 0,$$

which implies that

$$\frac{-\partial E_5}{\partial V_{ijkl}} = - \sum_{m=1}^{N_{sd}} \sum_{n=1}^{N_p(m)} \sum_{o=1}^{L(mn)} |A_{ijk} \cap A_{mno}| V_{mno l}.$$

Here, as in Section 4.1, A_{ijk} denotes the k^{th} link in the j^{th} path between SD pair i , and

$$|A_{ijk} \cap A_{mno}| = \begin{cases} 1, & \text{if } ijk \neq mno, \text{ and links } A_{ijk} \text{ and } A_{mno} \text{ share 1 or 2 nodes} \\ 0, & \text{if } ijk = mno, \text{ or links } A_{ijk} \text{ and } A_{mno} \text{ are disjoint} \end{cases}.$$

This is the first of the true scheduling constraints (see Section 4.1). Note that it prohibits the simultaneous activation of any adjacent links, including adjacent links in the same path, adjacent links in different paths between the same SD pair, and adjacent links in paths between different SD pairs.

RS_6 . Activate a total of N_x neurons:

The value of N_x is not known a priori, but since we have had success using $\beta < 1$ in the scheduling problem (which implies that the exact value of N_x is not critical in this constraint formulation), this constraint might be implemented as:

$$E_6 = \frac{1}{2} \left(\sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p(i)} \sum_{k=1}^{L(ij)} \sum_{l=1}^{\Lambda} V_{ijkl} - \beta N'_x \right)^2 = 0,$$

where N_x' may be set equal to an approximation of the number of required transmissions, such as the mean path length $\times N_{sd}$, or the sum of the shortest path lengths between each SD pair. Then, the corresponding equation-of-motion term is given by

$$\frac{-\partial E_6}{\partial V_{ijkl}} = \beta N_x' - \sum_{m=1}^{N_{sd}} \sum_{n=1}^{N_p^{(m)}} \sum_{o=1}^{L(mn)} \sum_{p=1}^{\Lambda} V_{mnop}.$$

RS7. Schedule all activations in slots numbered $\leq B_s$:

$$E_7 = \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p^{(i)}} \sum_{k=1}^{L(ij)} \sum_{l=1}^{\Lambda} W(l) V_{ijkl} = 0$$

which implies that

$$\frac{-\partial E_7}{\partial V_{ijkl}} = -W(l).$$

In the pure scheduling NN (Section 4.5.3), we used

$$W(l) = \begin{cases} (l - B_s)^2, & l > B_s \\ 0, & \text{otherwise} \end{cases},$$

where B_s is a known lower bound on the schedule length. However, in the joint routing-scheduling problem, we cannot readily calculate a near-tight bound because it is not known a priori which paths will be chosen. An estimated bound can be obtained by assuming that the shortest path between each of the SD pairs is selected, and calculating the schedule-length bound for the resultant set of paths, as described in Section 3.1. Alternatively, in this setting it might be better to use a function $W(l)$ that is not a function of the bound, but rather strictly a function of the slot, e.g., $W(l) = l$. The constraint would then no longer be an equality constraint (actually, it never was). As such, it should probably be viewed as an objective function rather than as a constraint.

RS8. Sequentially activate the links in each path (i.e., link ijk must be activated before link ijo , for $o > k$):

$$E_8 = \sum_{i=1}^{N_{sd}} \sum_{j=1}^{N_p^{(i)}} \sum_{k=1}^{L(ij)-1} \sum_{o=k+1}^{L(ij)} \sum_{l=1}^{\Lambda} \sum_{p=1}^{o-k+l-1} V_{ijkl} V_{ijop} = 0,$$

which implies that

$$\frac{-\partial E_8}{\partial V_{ijkl}} = -\left(1 - \delta_{kl(ij)}\right) \sum_{o=k+1}^{L(ij)} \sum_{p=1}^{o-k+l-1} V_{ijop} - (1 - \delta_{kl}) \sum_{o=1}^{k-1} \sum_{p=\max\{1, l-k+o+1\}}^{\Lambda} V_{ijop}.$$

This term provides a positive contribution to the energy function whenever two neurons that represent an out-of-sequence activation of the links in a path have nonzero output voltages. Since it represents purely inhibitory contributions to the connection weights, its impact in unselected paths can only be beneficial because it drives the neurons towards the desired zero state. In applications where it is not necessary to maintain the sequential order of link activation, the E_8 term is not included in the energy equation.

8.3 Alternate Models

The joint routing-scheduling NN model, besides requiring many more neurons than either the routing or the scheduling models, is also more heavily interconnected than either of the models that address the individual problems. Therefore, we address the question of whether the techniques we have developed to reduce the complexity of the pure scheduling model, by eliminating unnecessary neurons, can be applied to the joint problem as well.

The condensed NAS model presented in Section 4.5.1 is not applicable to the joint routing-NAS problem because in this problem it is necessary to maintain the correspondence between a link, its path, and its SD pair. Without specifying a single path between each SD pair, there is no way to determine a priori the number of times that a physical link must be activated ($N_x(p)$).

However, the reduced SAS model may be applicable to the joint routing-SAS problem. Application of this model would simply eliminate those neurons whose corresponding links cannot be activated in the corresponding time slot in an admissible sequential-activation schedule, as described in Section 4.5.2, thus achieving the desired reduction in NN complexity while also potentially removing from the solution space a number of local minima that represent inadmissible solutions.

9.0 CONCLUSIONS

In this report, we have addressed the use of Hopfield neural network (NN) models to solve the problem of link activation, or scheduling, in multihop packet radio networks. Both nonsequential-activation scheduling (NAS) and sequential-activation scheduling (SAS) models have been studied. A key feature of our models is the use of the method of Lagrange multipliers, which permits the connection weights to evolve dynamically along with the system state. This approach was also used in our earlier studies of Hopfield NN routing models [15, 16]. Other

important aspects of our models include the incorporation of heuristics into the equations of motion and the use of Gaussian simulated annealing, both of which encourage the evolution of the NN to optimal solutions.

Extensive simulation results have demonstrated the capability of our models to find optimal, i.e., minimum-length, schedules in many cases for which our heuristic (i.e., non-NN) approaches discussed in Section 3.2 were unable to do so. The degree of success obtained by the NN models is related to the degree to which the problem is constrained. For example, in some Monte-Carlo simulations of NAS problems, all of the solutions obtained from 100 different initial states (random seeds) were optimal. For a very highly-constrained problem, in which a typical schedule was 99.6% maximal, 21% of the solutions were optimal. Although this may appear to be a small number, it is notable that the NN model was able to determine optimal schedules for such a highly-constrained problem, whereas the aforementioned purely-heuristic approach was unable to do so. In studies of the SAS model, optimal solutions were found in as many as 79% of the runs.

Analysis of the sensitivity of our NN models to variations in parameter values and communication requirements has shown that both models are fairly robust. Both the NAS and the SAS models have produced optimal, or nearly optimal, schedules for a number of diverse problem instances without the need to adjust the parameters to accommodate different communication requirements. Simulations have shown that the performance of the NN does depend on the set of parameter values, but good performance is achieved over a broad range of these values. For example, one of the most critical parameters is the additional bias current parameter β . However, by means of a series of Monte-Carlo simulations of the NAS model, it was found that good performance (greater than 70% optimal solutions) was obtained with any value of β in the range [0.366, 1.122]; 100% of the schedules found by the NN were optimal when a value of β in the range [0.488, 0.854] was used.

In all of our NN models, the (expected) inability to guarantee a global optimum is mitigated by the fact that repeated runs are possible from different initial conditions; thus the best solution that is found can be chosen as the solution to the problem. As demonstrated by the multiple-instance simulation of Section 7.5.4 that scheduled the eight problem instances that are known to have minimum schedule lengths of 8 slots, the NN will often find an optimal schedule if allowed to relax from a sufficient number of different initial states. Although the simulation runs begin in random initial states, this method is not simply one of random search; system evolution is guided by the equations of motion, which are derived from the energy function, which in turn is based on the system constraints.

Our studies of routing and scheduling problems have verified our hypothesis that these two network control functions are not independent, and that schemes should be developed that jointly choose routes and link activation schedules. We have characterized the joint routing-scheduling problem as a combinatorial-optimization problem, and we have outlined the major components of a NN model for its solution. However, it would be difficult to simulate this model in software, except for very small networks, because of the large number of neurons and interconnections that are involved. It is hoped that future developments in the hardware design of NN components will be able to incorporate the techniques developed in this study to permit the solution of complicated communication network control problems of this type.

REFERENCES

1. D. J. Baker, A. Ephremides, and J. A. Flynn, "The Design and Simulation of a Mobile Radio Network with Distributed Control," *IEEE Journal on Selected Areas in Communications*, SAC-2 pp. 226 - 237, January 1984.
2. A. Ephremides, J. E. Wieselthier, and D. J. Baker, "A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling," *Proceedings of the IEEE*, 75-No. 1 pp. 56 - 73, January 1987.
3. M. J. Post, A. S. Kershenbaum, and P. E. Sarachik, "Scheduling Multihop CDMA Networks in the Presence of Secondary Conflicts," *Algorithmica*, 4 pp. 365 - 393, 1989.
4. J. E. Wieselthier, "Code-Division Multiple-Access Techniques and Their Application to the High-Frequency (HF) Intratask Force (ITF) Communication Network," NRL Report 9094, Naval Research Laboratory, Washington, D. C., September 1988.
5. E. Arıkan, "Some Complexity Results About Packet Radio Networks," *IEEE Transactions on Information Theory*, IT-30 pp. 681 - 685, July 1984.
6. B. Hajek and G. Sasaki, "Link Scheduling in Polynomial Time," *IEEE Transactions on Information Theory*, 34 pp. 910 - 917, September 1988.
7. L. Tassiulas, "Scheduling Problems in Multihop-Packet Radio Networks," Master's Thesis, University of Maryland, 1989.
8. U. Mukherji, "A Periodic Scheduling Problem in Flow Control for Data Communication Networks," *IEEE Transactions on Information Theory*, 35-No. 2 pp. 436 - 443, March 1989.
9. L. Tassiulas and A. Ephremides, "An Algorithm for Joint Routing and Scheduling in Radio Networks," *Proceedings of the American Control Conference*, pp. 697 - 702, June 1989.
10. J. J. Hopfield and D. W. Tank, "'Neural' Computation of Decisions in Optimization Problems," *Biological Cybernetics*, 52 pp. 141 - 152, 1985.

11. L. Tassiulas, A. Ephremides, and J. Gunn, "Solving Hard Optimization Problems Arising in Packet Radio Networks Using Hopfield's Net," *Proceedings of the 1989 Conference on Information Sciences and Systems*, pp. 603 - 608, March 1989.
12. C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "Neural Network Techniques for Scheduling and Routing Problems in Multihop Radio Networks," to appear in *The MILCOM'91 Conference Proceedings*, November 1991.
13. J. E. Wieselthier, C. M. Barnhart, and A. Ephremides, "Sequential Link Activation in Multihop Radio Networks by Means of Hopfield Neural Network Techniques," *Proceedings of the 1991 International Symposium on Information Theory*, p. 155, June 1991.
14. C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "The Use of Hopfield Neural Nets in Combinatorial-Optimization Problems Arising in Radio Communication Networks," *The Thirtieth International Meeting of the Institute of Management Sciences (TIMS)*, July 1991.
15. J. E. Wieselthier, C. M. Barnhart, and A. Ephremides, "The Application of Hopfield Neural Network Techniques to Problems of Routing and Scheduling in Packet Radio Networks," NRL Memorandum Report 6730, Naval Research Laboratory, Washington D. C., November 1990.
16. J. E. Wieselthier, C. M. Barnhart, and A. Ephremides, "A Neural Network Approach to Routing in Multihop Radio Networks," *Proceedings of IEEE INFOCOM'91*, pp. 1074 - 1083, April 1991.
17. J. E. Wieselthier, C. Barnhart, A. Ephremides, and W. Thoet, "Routing and Scheduling in Packet Radio Networks: A Hopfield Network Approach," *Proceedings of the 1990 Conference on Information Sciences and Systems*, p. 185, March 1990.
18. R. G. Ogier, "A Decomposition Method for Optimal Link Scheduling," *Proceedings of the 24th Allerton Conference*, October 1986.
19. M. J. Post, P. E. Sarachik, and A. S. Kershenbaum, "A 'Biased Greedy' Algorithm for Scheduling Multi-Hop Radio Networks," *Proceedings of the '85 Conference on Information Science and Systems*, pp. 564 - 572, March 1985.
20. C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Englewood Cliffs: Prentice-Hall, Inc., 1982.
21. J. C. Platt and A. H. Barr, "Constrained Differential Optimization," *Proceedings of the IEEE 1987 Neural Information Processing Systems Conference*, pp. 612 - 621, 1987.
22. E. Wacholder, J. Han, and R. C. Mann, "A Neural Network Algorithm for the Multiple Traveling Salesmen Problem," *Biological Cybernetics*, 61 pp. 11 - 19, 1989.
23. P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*, Dordrecht: D. Reidel Publishing Company, 1987.
24. S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, 220 pp. 671 - 680, May 13, 1983.

25. B. Hajek, "A Tutorial Survey of Theory and Applications of Simulated Annealing," *Proceedings of the 24th IEEE Conference on Decision and Control*, pp. 755 - 760, December 1985.
26. Y. Akiyama, A. Yamashita, M. Kajiura, and H. Aiso, "Combinatorial Optimization with Gaussian Machines," *Proceedings of the International Joint Conference on Neural Networks*, pp. I-533 - I-540, 1989.
27. Y. Akiyama, A. Yamashita, M. Kajiura, Y. Anzai, and H. Aiso, "The Gaussian Machine: A Stochastic Neural Network for Solving Assignment Problems," *Journal of Neural Network Computing*, 2-No. 3 pp. 43 - 51, Winter 1991.
28. D. E. Van den Bout and T. K. Miller III, "Graph Partitioning Using Annealed Neural Networks," *IEEE Transactions on Neural Networks*, 1 pp. 192 - 203, June 1990.
29. A. S. Tanenbaum, *Computer Networks, Second Edition*, Englewood Cliffs, NJ: Prentice Hall, 1988.

APPENDIX A

TABLES OF THE PATHS AND PROBLEM INSTANCES ASSOCIATED WITH THE NETWORK OF FIGURE 2.1

Table A1. Listing of paths for the 24-node network of Fig. 2.1									
SD pair	Path	Nodes Traversed							
1	0	4	5	13	20	24			
	1	4	7	14	19	20	24		
	2	4	3	6	11	12	13	20	24
	3	4	7	13	20	24			
	4	4	5	14	19	20	24		
	5	4	3	6	8	9	12	13	20 24
2	6	7	14	15	17				
	7	7	11	12	13	19	14	15	17
3	8	9	11	7	14	15	16		
	9	9	8	6	5	14	15	16	
	10	9	12	13	19	14	15	16	
	11	9	12	7	14	15	16		
4	12	1	4	5	13	19			
	13	1	2	3	6	7	14	19	
	14	1	4	7	14	19			
	15	1	2	3	6	11	12	13	19
	16	1	4	5	14	19			
	17	1	4	7	13	19			
	18	1	2	3	6	8	9	12	13 19
	19	1	2	3	6	5	14	19	
5	20	5	6	11					
	21	5	7	11					
	22	5	13	12	11				
6	23	21	22	20	13	5	6		
	24	21	24	20	19	14	7	6	
	25	21	22	20	13	12	11	6	
	26	21	24	20	13	5	6		
	27	21	22	20	19	14	7	6	
	28	21	24	20	13	12	9	8	6
	29	21	24	20	13	7	6		
	30	21	22	20	19	14	5	6	
	31	21	24	20	13	12	11	6	
	32	21	22	20	13	7	6		
	33	21	24	20	19	14	5	6	
	34	21	22	20	13	12	9	8	6
7	35	1	4	7	11	10			
	36	1	2	3	6	8	9	10	
	37	1	4	5	13	12	9	10	
	38	1	2	3	6	11	10		

Table A1. (continued)										
SD pair	Path	Nodes Traversed								
8	39	3	4	5	14	15	18			
	40	3	6	7	14	15	18			
	41	3	6	11	12	13	19	14	15	18
	42	3	4	7	14	15	18			
	43	3	6	5	14	15	18			
	44	3	6	8	9	12	13	19	14	15 18
9	45	2	4	7	12					
	46	2	3	6	11	12				
	47	2	4	5	13	12				
	48	2	3	6	8	9	12			
10	49	14	5	6	8					
	50	14	7	11	8					
	51	14	19	13	12	9	8			

A Heavy Set of Communication Demands (Section 6.3)

The “augmented” problem instance listed in Table A2 consists of the paths connecting the 10 SD pair listed in Table 2.1 (Section 2.3) plus a set of paths that connects an additional 14 SD pairs. The additional paths add no physical links to the network, but 22 more transmissions than are required by the original problem of Table 2.1 are needed to satisfy the demand of Table A2. As a starting point for the generation of the communication requirements for the augmented problem, we considered one of the 8-slot schedules for the original problem. We then added a set of additional link activations (and hence their corresponding paths) so that each slot in the resulting schedule would be maximally scheduled.¹ The schedule generated by the NAS heuristic for this problem is 9 slots long, and the minimum nonsequential-activation schedule length is 8 slots.

Table A2. A heavy communication demand

	Path no. [†]	Path (nodes traversed)					
SD pair 1: [4, 24]	0	4	5	13	20	24	
SD pair 2: [7, 17]	6	7	14	15	17		
SD pair 3: [9, 16]	10	9	12	13	19	14	15 16
SD pair 4: [1, 19]	12	1	4	5	13	19	
SD pair 5: [5, 11]	20	5	6	11			
SD pair 6: [21, 6]	23	21	22	20	13	5	6
SD pair 7: [1, 10]	36	1	2	3	6	8	9 10
SD pair 8: [3, 18]	42	3	4	7	14	15	18
SD pair 9: [2, 12]	45	2	4	7	12		
SD pair 10: [14, 8]	50	14	7	11	8		
SD pair 11: [1, 4]		1	2	4			
SD pair 12: [1, 3]		1	2	3			
SD pair 13: [2, 3]		2	3				
SD pair 14: [16, 18]		16	15	18			
SD pair 15: [8, 10]		8	9	10			
SD pair 16: [10, 8]		10	9	8			
SD pair 17: [8, 9]		8	9				
SD pair 18: [7, 11]		7	11				
SD pair 19: [21, 20]		21	22	20			
SD pair 20: [20, 21]		20	22	21			
SD pair 21: [22, 20]		22	20				
SD pair 22: [20, 22]		20	22				
SD pair 23: [5, 11]	20	5	6	11			
SD pair 24: [19, 14]		19	14				

[†] See Table A1. SD pairs 11 through 22, and 24 are not listed in Table A1. Although the path connecting SD pair 23 is identical to that connecting SD pair 5, the two paths are considered to be distinct entities.

¹ Although the communication requirements for the augmented problem are based on a maximal schedule, nonmaximal schedules (in which one or more additional links can be activated in one or more slots) satisfying the same communication requirements also exist.

A Third Set of Communication Requirements (Section 6.3.1)

Table A3. A communication demand requiring 44 transmissions in its 28 physical links (minimum nonsequential-activation schedule length is 7 slots, NAS heuristic schedule length is 8 slots).

	Path no. [†]	Path (nodes traversed)									
SD pair 1: [4, 24]	0	4	5	13	20	24					
SD pair 2: [7, 17]	6	7	14	15	17						
SD pair 3: [9, 16]	11	9	12	7	14	15	16				
SD pair 4: [1, 19]	18	1	2	3	6	8	9	12	13	19	
SD pair 5: [5, 11]	20	5	6	11							
SD pair 6: [21, 6]	28	21	24	20	13	12	9	8	6		
SD pair 7: [1, 10]	35	1	4	7	11	10					
SD pair 8: [3, 18]	39	3	4	5	14	15	18				
SD pair 9: [2, 12]	45	2	4	7	12						
SD pair 10: [14, 8]	49	14	5	6	8						

[†] See Table A1.

Sequences of Problem Instances

In the following three tables, sequences of problem instances are listed. Rather than enumerating each of the nodes traversed in every path for each problem instance, the ten paths in each of the problem instances are listed by their path numbers, which correspond to the paths listed in Table A1. The nodes traversed in each of these numbered paths are listed in Table A1.

Table A4. The set of 8 problem instances with $B_{sas}=8$ and SAS heuristic schedule lengths of 8 slots (Section 7.5.1).

Path Set #	SD Pair (path number [†])									
	1	2	3	4	5	6	7	8	9	10
1	3	6	11	12	22	25	36	39	45	49
2	3	6	11	12	22	31	36	39	45	49
3	3	6	9	12	22	25	36	42	45	49
4	3	6	9	12	22	31	36	42	45	49
5	3	6	9	12	22	28	38	42	45	49
6	3	6	9	12	22	34	38	42	45	49
7	0	6	11	12	22	29	36	43	48	50
8	0	6	11	12	22	32	36	43	48	50

[†] See Table A1.

Table A5. A partial listing of set (7, >7) ($B_{nas} = 7$, NAS heuristic schedule length > 7 slots, Sections 6.5 and 7.5)

Path Set #	SD Pair (path number, see Table A1)									
	1	2	3	4	5	6	7	8	9	10
1	0	6	11	18	20	28	35	39	45	49
2	2	6	11	12	21	25	36	39	45	49
3	5	6	11	12	21	25	36	39	45	49
4	0	6	8	15	21	25	36	39	45	49
5	0	6	11	15	21	25	36	39	45	49
6	0	6	8	18	21	25	36	39	45	49
7	0	6	11	18	21	25	36	39	45	49
8	2	6	11	12	21	28	36	39	45	49
9	5	6	11	12	21	28	36	39	45	49
10	0	6	11	15	21	28	36	39	45	49
11	0	6	8	18	21	28	36	39	45	49
12	5	6	11	12	21	31	36	39	45	49
13	0	6	8	15	21	31	36	39	45	49
14	0	6	11	15	21	31	36	39	45	49
15	0	6	8	18	21	31	36	39	45	49
16	0	6	11	18	21	31	36	39	45	49
17	5	6	8	12	21	34	36	39	45	49
18	2	6	11	12	21	34	36	39	45	49
19	5	6	11	12	21	34	36	39	45	49
20	0	6	8	15	21	34	36	39	45	49
21	0	6	11	15	21	34	36	39	45	49
22	0	6	11	18	21	34	36	39	45	49
23	0	6	11	15	21	25	38	39	45	49
24	0	6	8	18	21	25	38	39	45	49
25	0	6	11	18	21	25	38	39	45	49
26	0	6	8	15	21	28	38	39	45	49
27	0	6	11	15	21	28	38	39	45	49
28	0	6	8	18	21	28	38	39	45	49
29	0	6	11	18	21	28	38	39	45	49
30	0	6	11	15	21	31	38	39	45	49
31	0	6	8	18	21	31	38	39	45	49
32	0	6	11	18	21	31	38	39	45	49
33	5	6	8	12	21	34	38	39	45	49
34	0	6	8	15	21	34	38	39	45	49
35	0	6	11	15	21	34	38	39	45	49
36	0	6	8	18	21	34	38	39	45	49
37	0	6	11	18	21	25	35	39	46	49
38	3	6	8	12	20	28	35	39	46	49
39	0	6	8	17	20	28	35	39	46	49
40	5	6	11	12	21	28	35	39	46	49
41	0	6	8	18	21	28	35	39	46	49
42	0	6	11	18	21	28	35	39	46	49
43	0	6	11	18	21	31	35	39	46	49
44	3	6	8	12	20	34	35	39	46	49
45	0	6	8	17	20	34	35	39	46	49
46	5	6	11	12	21	34	35	39	46	49
47	0	6	8	18	21	34	35	39	46	49
48	0	6	11	18	21	34	35	39	46	49
49	3	6	8	12	21	25	36	39	46	49
50	3	6	11	12	21	25	36	39	46	49

Table A6. A partial listing of set (7, 7) ($B_{nas} = 7$, NAS heuristic schedule length = 7 slots, Section 6.5)

Path Set #	SD Pair (path number, see Table A1)									
	1	2	3	4	5	6	7	8	9	10
1	0	6	11	15	20	25	35	39	45	49
2	0	6	8	18	20	25	35	39	45	49
3	0	6	11	18	20	25	35	39	45	49
4	0	6	8	15	20	28	35	39	45	49
5	0	6	11	15	20	28	35	39	45	49
6	0	6	8	18	20	28	35	39	45	49
7	0	6	11	15	20	31	35	39	45	49
8	0	6	8	18	20	31	35	39	45	49
9	0	6	11	18	20	31	35	39	45	49
10	0	6	8	15	20	34	35	39	45	49
11	0	6	11	15	20	34	35	39	45	49
12	0	6	8	18	20	34	35	39	45	49
13	0	6	11	18	20	34	35	39	45	49
14	3	6	8	12	20	25	36	39	45	49
15	3	6	11	12	20	25	36	39	45	49
16	0	6	8	17	20	25	36	39	45	49
17	0	6	11	17	20	25	36	39	45	49
18	2	6	8	12	21	25	36	39	45	49
19	5	6	8	12	21	25	36	39	45	49
20	3	6	8	12	20	28	36	39	45	49
21	3	6	11	12	20	28	36	39	45	49
22	0	6	8	17	20	28	36	39	45	49
23	0	6	11	17	20	28	36	39	45	49
24	2	6	8	12	21	28	36	39	45	49
25	5	6	8	12	21	28	36	39	45	49
26	0	6	8	15	21	28	36	39	45	49
27	0	6	11	18	21	28	36	39	45	49
28	3	6	8	12	20	31	36	39	45	49
29	3	6	11	12	20	31	36	39	45	49
30	0	6	8	17	20	31	36	39	45	49
31	0	6	11	17	20	31	36	39	45	49
32	2	6	8	12	21	31	36	39	45	49
33	5	6	8	12	21	31	36	39	45	49
34	2	6	11	12	21	31	36	39	45	49
35	3	6	8	12	20	34	36	39	45	49
36	3	6	11	12	20	34	36	39	45	49
37	0	6	8	17	20	34	36	39	45	49
38	0	6	11	17	20	34	36	39	45	49
39	2	6	8	12	21	34	36	39	45	49
40	0	6	8	18	21	34	36	39	45	49
41	3	6	8	12	20	25	38	39	45	49
42	3	6	11	12	20	25	38	39	45	49
43	0	6	8	17	20	25	38	39	45	49
44	0	6	11	17	20	25	38	39	45	49
45	5	6	8	12	21	25	38	39	45	49
46	2	6	11	12	21	25	38	39	45	49
47	5	6	11	12	21	25	38	39	45	49
48	3	6	8	12	20	28	38	39	45	49
49	3	6	11	12	20	28	38	39	45	49
50	0	6	8	17	20	28	38	39	45	49

APPENDIX B

TIGHTENING THE SAS BOUND (AN EXAMPLE)

For several of the problem instances evaluated in this report, we have been able to tighten the sequential-activation schedule length lower bound B_{sas} by contradiction. That is, we assume that an admissible sequential-activation schedule of length B_{sas} does exist. Then, in the process of trying to find such a schedule, we show that an admissible B_{sas} -slot schedule cannot exist. Therefore, the minimum sequential-activation schedule length Λ^* must be greater than B_{sas} , and the bound can be tightened by one.

For example, the problem instance given by Table A3 in Appendix A has $B_{sas} = 8$ slots. Table B1 shows an effort to find an 8-slot sequential-activation schedule for this problem instance. As can be seen in the table, the path connecting the fourth SD pair is 8 hops in length. Therefore, to maintain the sequentiality requirement in an 8-slot schedule, the first link of this path must be activated in the first slot, the second link must be activated in the second slot, and so on. Thus, we enter "X" in the eight cells as shown in the table. These now-scheduled link activations block the activation of certain other links that share nodes with the activated links in the assigned slot. We enter "b" in these cells that would result in a primary conflict if their associated link were activated in that slot. Note that this results in a complete blockage of link 6,5 (the fifth link in the path between SD pair 6). Since there is no slot in which link 6,5 may be legally activated in an 8-slot schedule, the minimum sequential-activation schedule length for this problem must be greater than 8 slots. Therefore, we may safely say that a tightened lower bound for this problem is 9 slots. It turns out that $\Lambda^* = 9$ slots, as verified by the existence of a feasible 9-slot schedule found by the reduced SAS NN model for this problem.

Table B1. An effort find an 8-slot sequential-activation schedule for the requirements of Table A3

Path†	Link	Indices (SD, link)	Slot							
			1	2	3	4	5	6	7	8
0	(4->5)	1,1	o	o	o	o	o			
	(5->13)	1,2		o	o	o	o	o		
	(13->20)	1,3			o	o	o	o	b	
	(20->24)	1,4				o	o	o	o	o
6	(7->14)	2,1	o	o	o	o	o	o		
	(14->15)	2,2		o	o	o	o	o	o	
	(15->17)	2,3			o	o	o	o	o	o
11	(9->12)	3,1	o	o	o	o				
	(12->7)	3,2		o	o	o	o			
	(7->14)	3,3			o	o	o	o		
	(14->15)	3,4				o	o	o	o	
	(15->16)	3,5					o	o	o	o
18	(1->2)	4,1	X							
	(2->3)	4,2		X						
	(3->6)	4,3			X					
	(6->8)	4,4				X				
	(8->9)	4,5					X			
	(9->12)	4,6						X		
	(12->13)	4,7							X	
	(13->19)	4,8								X
20	(5->6)	5,1	o	o	b	b	o	o	o	
	(6->11)	5,2		o	b	b	o	o	o	o
28	(21->24)	6,1	o	o						
	(24->20)	6,2		o	o					
	(20->13)	6,3			o	o				
	(13->12)	6,4				o	o			
	(12->9)	6,5					b	b		
	(9->8)	6,6						b	o	
	(8->6)	6,7							o	o
35	(1->4)	7,1	b	o	o	o	o			
	(4->7)	7,2		o	o	o	o	o		
	(7->11)	7,3			o	o	o	o	o	
	(11->10)	7,4				o	o	o	o	o
39	(3->4)	8,1	o	b	b	o				
	(4->5)	8,2		o	o	o	o			
	(5->14)	8,3			o	o	o	o		
	(14->15)	8,4				o	o	o	o	
	(15->18)	8,5					o	o	o	o
45	(2->4)	9,1	b	b	o	o	o	o		
	(4->7)	9,2		o	o	o	o	o	o	
	(7->12)	9,3			o	o	o	b	b	o
49	(14->5)	10,1	o	o	o	o	o	o		
	(5->6)	10,2		o	b	b	o	o	o	
	(6->8)	10,3			b	b	b	o	o	o

† See Table A.1 in Appendix A.